Atmel mXT2952T2 Revision 1.1

maXTouch 2911-node Touchscreen Controller

DATASHEET

Features

- Atmel® maXTouch® Adaptive Sensing Touchscreen Technology
  - Up to 41 X (transmit) lines and 71 Y (receive) lines
  - A maximum of 2911 nodes can be allocated to the touchscreen
  - Screen sizes of 11.6 - 15.6 inches diagonal are supported
  - Multi-touch support with up to 16 concurrent touches tracked in real time
- Dual-boot OS support for Windows and Android
- Advanced Touch Handling
  - Moisture/Water Compensation
    - No false touch with condensation or water drop up to 22 mm diameter
    - One-finger tracking with condensation or water drop up to 22 mm diameter
  - Stylus Support
    - Supports passive stylus with 1 mm contact diameter, subject to configuration, stack up, and sensor design
  - Glove Support
    - Supports multiple-finger glove touch up to 1.5 mm thickness
    - Supports single-touch gloved operation with various materials up to 5 mm thickness
- Touch Performance
  - Mutual capacitance and self capacitance measurements supported for touch detection
  - Response Times
    - Initial latency <10 ms for first touch from idle, subject to configuration
  - Atmel maXCharger® technology to combat ambient, charger noise, and power-line noise:
    - Up to 240 Vpp between 1 Hz and 1 kHz sinusoidal waveform
    - Up to 20 Vpp between 1 kHz and 1 MHz sinusoidal waveform
  - Scan Speed
    - Typical report rate for 15 touches ≥100 Hz
- Enhanced Algorithms
  - Lens bending algorithms to remove signal distortions
  - Touch suppression algorithms to remove unintentional touches
  - Palm Recovery Algorithm for quick restoration to normal state
- Panel / Cover Glass Support
  - Supports fully-laminated sensors, touch-on-lens stack-ups and on-cell designs
  - Works with PET or glass, including curved profiles
    - Glass from 0.55 to 2.5 mm, dependent on screen size and touch size
    - Plastic from 0.2 mm to 1.2 mm, dependent on screen size and touch size
  - Works with all proprietary sensor patterns recommended by Atmel
  - Compatible with True Single Layer designs

- Keys
  - Up to 64 nodes can be allocated as mutual capacitance sensor keys (subject to other configurations)
  - Adjacent Key Suppression® (AKS®) technology is supported for false touch prevention
- Power Saving
  - Programmable timeout for automatic transition from active to idle states
  - Pipelined analog sensing detection and digital processing to optimize system power efficiency
- Application Interfaces
  - I^2C-compatible slave mode: Standard/Fast mode 400 kHz, Fast-plus mode 1 MHz, High-speed mode up to 3.4 MHz
  - USB composite device, full speed (12 Mbps)
  - HID-I^2C interface for Microsoft® Windows® 8.x
  - Interrupt to indicate when a message is available
- Power Supply
  - Digital (Vdd) 3.3 V nominal
  - Analog (AVdd) 3.3 V nominal
  - Host interface I/O voltage (VddIO) 1.8 V to 3.3 V nominal
  - High voltage external X line drive (XVdd) 9.0 V maximum (1)
- Package
  - 162-ball UFBGA 10 × 5 × 0.6 mm, 0.5 mm pitch
- Environmental Conditions
  - Operating temperature –40°C to +85°C

1. This is an Absolute Maximum value – the Nominal value must be set lower to allow for component tolerances.

Table of Contents

1. Overview of mXT2952T2

1.1 Introduction

The Atmel maXTouch family of touch controllers brings industry-leading capacitive touch performance to customer applications. The mXT2952T2 features the latest generation of Atmel Adaptive Sensing technology that utilizes a hybrid mutual- and self-capacitive sensing system in order to deliver unparalleled touch features and a robust user experience.

- **Patented capacitive sensing method** – The mXT2952T2 uses a unique charge-transfer acquisition engine to implement the Atmel-patented QMatrix® capacitive sensing method. Coupled with a state-of-the-art CPU, the entire touchscreen sensing solution can measure, classify and track number of individual finger touches with a high degree of accuracy in the shortest response time.
- **Capacitive Touch Engine (CTE)** – The mXT2952T2 features an acquisition engine, which uses an optimal measurement approach to ensure almost complete immunity from parasitic capacitance on the receiver input lines. The engine includes sufficient dynamic range to cope with anticipated touchscreen self and mutual capacitances, which allows great flexibility for use with the Atmel proprietary sensor pattern designs. One- and two-layer ITO sensors are possible using glass or PET substrates.
- **Touch detection** – The mXT2952T2 allows for both mutual- and self-capacitance measurements, with the self-capacitance measurements being used to augment the mutual-capacitance measurements to produce reliable touch information.

  When self-capacitance measurements are enabled, touch classification is achieved using both mutual- and self-capacitance touch data. This has the advantage that both types of measurement systems can work together to detect touches under a wide variety of circumstances.

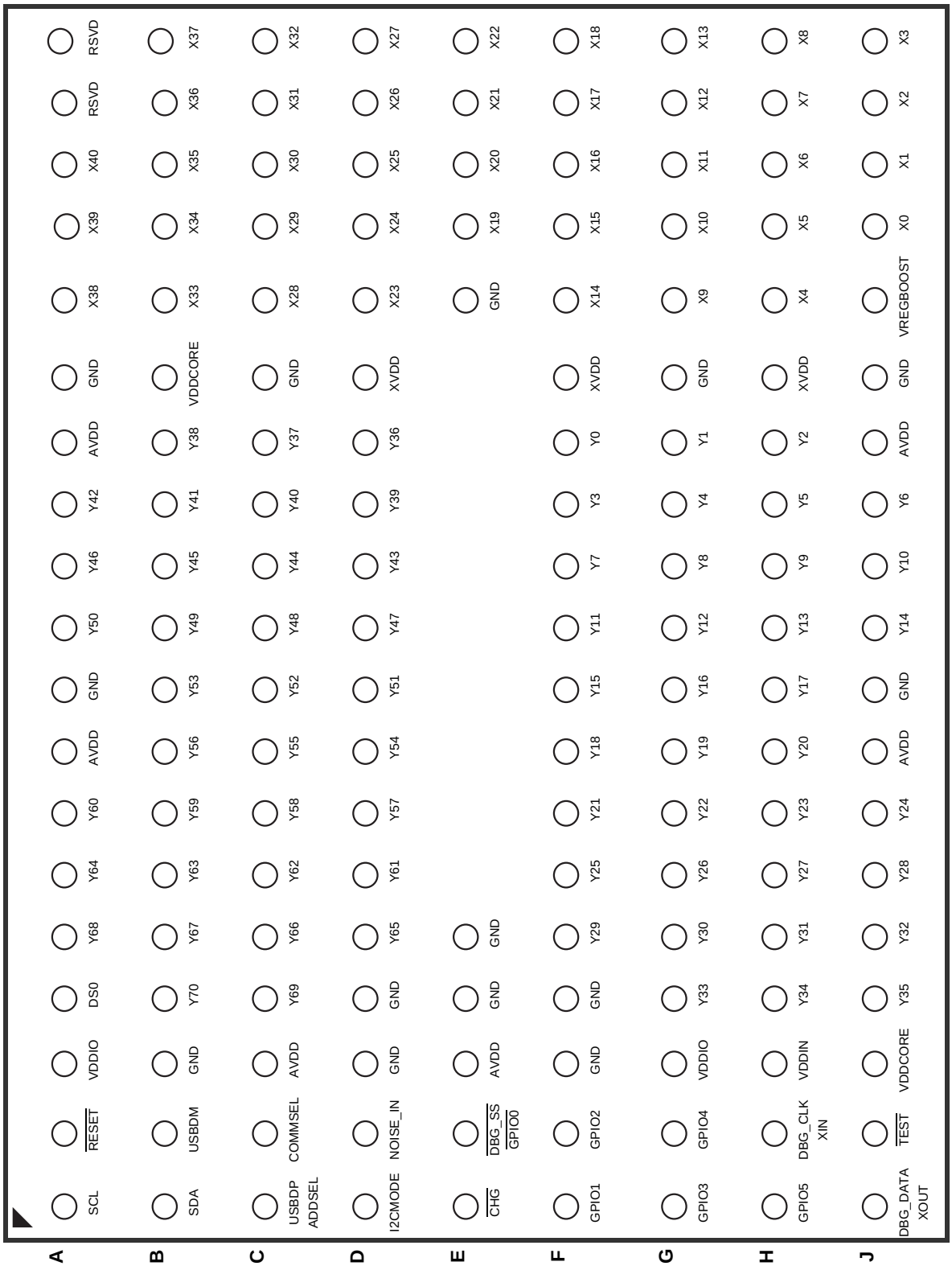
  During idle mode, the device performs self-capacitance touch scans. When a touch is detected, the device starts performing mutual-capacitance touch scans as well as self capacitance scans.

  Mutual-capacitance touch data is used wherever possible to classify touches as this has greater granularity than self-capacitance measurements and provides positional information on touches. For this reason, multiple touches can only be determined by mutual-capacitance touch data. If the self-capacitance touch processing detects multiple touches, touchscreen processing is skipped until mutual-capacitance touch data is available.

  Self-capacitance measurements, on the other hand, allow for the detection of single touches in extreme case, such as single thick-glove touches, when touches can only be detected by self-capacitance data and may be missed by mutual-capacitance touch detection.
- **Display Noise Cancellation** – A combination of analog circuitry, hardware noise processing, and firmware that combats display noise without requiring additional listening channels or synchronization to display timing. This enables the use of shieldless touch sensor stacks, including touch-on-lens.
- **Noise filtering** – Hardware noise processing in the capacitive touch engine provides enhanced autonomous filtering and allows a broad range of noise profiles to be handled. The result is good performance in the presence of charger and LCD noise.
- **Processing power** – The main CPU has two powerful microsequencer coprocessors under its control consuming low power. This system allows the signal acquisition, preprocessing, postprocessing and housekeeping to be partitioned in an efficient and flexible way.
- **Interpreting user intention** – The Atmel hybrid mutual- and self-capacitance method provides unambiguous multitouch performance. Algorithms in the mXT2952T2 provide optimized touchscreen position filtering for the smooth tracking of touches, responding to a user's intended touches while preventing false touch triggered by ambient noise or conductive material on the sensor surface, such as water. The suppression of unintentional touches from the user's gripping fingers, resting palm or touching cheek or ear also help ensure that the user's intentions are correctly interpreted.

2. Connection and Configuration Information

2.1 Pin Configuration – UFBGA 162 Balls



Top View

Table 2-1. Pin Listing – UFBGA 162 Balls

Pin	Name	Type	Description	If Unused...
A1	SCL	OD	Serial clock input	Connect to VDD (USB Mode)
A2	$\overline{\text{RESET}}$	I	Reset low. Connection to host system is recommended	Pull up to VddIO
A3	VDDIO	P	Digital power	–
A4	DS0	O	Driven Shield signal; used as guard track between X/Y signals and ground	Leave open
A5	Y68	S	Y line connection	Leave open
A6	Y64	S	Y line connection	Leave open
A7	Y60	S	Y line connection	Leave open
A8	AVDD	P	Analog power	–
A9	GND	P	Ground	–
A10	Y50	S	Y line connection	Leave open
A11	Y46	S	Y line connection	Leave open
A12	Y42	S	Y line connection	Leave open
A13	AVDD	P	Analog power	–
A14	GND	P	Ground	–
A15	X38	S	X matrix drive line	Leave open
A16	X39	S	X matrix drive line	Leave open
A17	X40	S	X matrix drive line	Leave open
A18	RSVD	O	Reserved	Leave open
A19	RSVD	O	Reserved	Leave open
B1	SDA	OD	Serial interface data	Pull up to VddIO (USB Mode)
B2	USBDM	USB	USB device port minus	Connect to GND
B3	GND	P	Ground	–
B4	Y70	S	Y line connection	Leave open
B5	Y67	S	Y line connection	Leave open
B6	Y63	S	Y line connection	Leave open
B7	Y59	S	Y line connection	Leave open
B8	Y56	S	Y line connection	Leave open
B9	Y53	S	Y line connection	Leave open
B10	Y49	S	Y line connection	Leave open
B11	Y45	S	Y line connection	Leave open

Table 2-1. Pin Listing – UFBGA 162 Balls (Continued)

Pin	Name	Type	Description	If Unused...
B12	Y41	S	Y line connection	Leave open
B13	Y38	S	Y line connection	Leave open
B14	VDDCORE	P	Digital core power	–
B15	X33	S	X matrix drive line	Leave open
B16	X34	S	X matrix drive line	Leave open
B17	X35	S	X matrix drive line	Leave open
B18	X36	S	X matrix drive line	Leave open
B19	X37	S	X matrix drive line	Leave open
C1	USBDP ADDSEL	I/O	USB data port plus I2C address select: See Section 9.3 on page 32	Leave open (USB mode). Pull up/down as required (I2C mode).
C2	COMMSEL	I	Communications interface selection: See Section 9.1 on page 31	–
C3	AVDD	P	Analog power	–
C4	Y69	S	Y line connection	Leave open
C5	Y66	S	Y line connection	Leave open
C6	Y62	S	Y line connection	Leave open
C7	Y58	S	Y line connection	Leave open
C8	Y55	S	Y line connection	Leave open
C9	Y52	S	Y line connection	Leave open
C10	Y48	S	Y line connection	Leave open
C11	Y44	S	Y line connection	Leave open
C12	Y40	S	Y line connection	Leave open
C13	Y37	S	Y line connection	Leave open
C14	GND	P	Ground	–
C15	X28	S	X matrix drive line	Leave open
C16	X29	S	X matrix drive line	Leave open
C17	X30	S	X matrix drive line	Leave open
C18	X31	S	X matrix drive line	Leave open
C19	X32	S	X matrix drive line	Leave open
D1	I2CMODE	I	Selects I2C mode: See Section 9.2 on page 31	–
D2	NOISE_IN	I	Charger present input	Connect to GND
D3	GND	P	Ground	–
D4	GND	P	Ground	–

Table 2-1. Pin Listing – UFBGA 162 Balls (Continued)

Pin	Name	Type	Description	If Unused...
D5	Y65	S	Y line connection	Leave open
D6	Y61	S	Y line connection	Leave open
D7	Y57	S	Y line connection	Leave open
D8	Y54	S	Y line connection	Leave open
D9	Y51	S	Y line connection	Leave open
D10	Y47	S	Y line connection	Leave open
D11	Y43	S	Y line connection	Leave open
D12	Y39	S	Y line connection	Leave open
D13	Y36	S	Y line connection	Leave open
D14	XVDD	P	X line drive power	–
D15	X23	S	X matrix drive line	Leave open
D16	X24	S	X matrix drive line	Leave open
D17	X25	S	X matrix drive line	Leave open
D18	X26	S	X matrix drive line	Leave open
D19	X27	S	X matrix drive line	Leave open
E1	$\overline{\text{CHG}}$	OD	Change line interrupt	Pull up to VddIO
E2	$\overline{\text{DBG\_SS}}$ GPIO0	I/O	Debug SS line; pull up to VddIO General purpose IO	Input: Connect to GND Output: Leave open
E3	AVDD	P	Analog power	–
E4	GND	P	Ground	–
E5	GND	P	Ground	–
...				
E15	GND	P	Ground power	–
E16	X19	S	X matrix drive line	Leave open
E17	X20	S	X matrix drive line	Leave open
E18	X21	S	X matrix drive line	Leave open
E19	X22	S	X matrix drive line	Leave open
F1	GPIO1	I/O	General purpose IO	Input: Connect to GND Output: Leave open
F2	GPIO2	I/O	General purpose IO	Input: Connect to GND Output: Leave open
F3	GND	P	Ground	–
F4	GND	P	Ground	–

Table 2-1. Pin Listing – UFBGA 162 Balls (Continued)

Pin	Name	Type	Description	If Unused...
F5	Y29	S	Y line connection	Leave open
F6	Y25	S	Y line connection	Leave open
F7	Y21	S	Y line connection	Leave open
F8	Y18	S	Y line connection	Leave open
F9	Y15	S	Y line connection	Leave open
F10	Y11	S	Y line connection	Leave open
F11	Y7	S	Y line connection	Leave open
F12	Y3	S	Y line connection	Leave open
F13	Y0	S	Y line connection	Leave open
F14	XVDD	P	X line drive power	–
F15	X14	S	X matrix drive line	Leave open
F16	X15	S	X matrix drive line	Leave open
F17	X16	S	X matrix drive line	Leave open
F18	X17	S	X matrix drive line	Leave open
F19	X18	S	X matrix drive line	Leave open
G1	GPIO3	I/O	General purpose IO	Input: Connect to GND Output: Leave open
G2	GPIO4	I/O	General purpose IO	Input: Connect to GND Output: Leave open
G3	VDDIO	P	Host interface power	–
G4	Y33	S	Y line connection	Leave open
G5	Y30	S	Y line connection	Leave open
G6	Y26	S	Y line connection	Leave open
G7	Y22	S	Y line connection	Leave open
G8	Y19	S	Y line connection	Leave open
G9	Y16	S	Y line connection	Leave open
G10	Y12	S	Y line connection	Leave open
G11	Y8	S	Y line connection	Leave open
G12	Y4	S	Y line connection	Leave open
G13	Y1	S	Y line connection	Leave open
G14	GND	P	Ground	–
G15	X9	S	X matrix drive line	Leave open
G16	X10	S	X matrix drive line	Leave open

Table 2-1. Pin Listing – UFBGA 162 Balls (Continued)

Pin	Name	Type	Description	If Unused...
G17	X11	S	X matrix drive line	Leave open
G18	X12	S	X matrix drive line	Leave open
G19	X13	S	X matrix drive line	Leave open
H1	GPIO5	I/O	General purpose IO	Input: Connect to GND Output: Leave open
H2	DBG_CLK XIN	O I	Debug clock External oscillator input	Output: Leave open Input: Connect to GND
H3	VDDIN	P	Digital power	–
H4	Y34	S	Y line connection	Leave open
H5	Y31	S	Y line connection	Leave open
H6	Y27	S	Y line connection	Leave open
H7	Y23	S	Y line connection	Leave open
H8	Y20	S	Y line connection	Leave open
H9	Y17	S	Y line connection	Leave open
H10	Y13	S	Y line connection	Leave open
H11	Y9	S	Y line connection	Leave open
H12	Y5	S	Y line connection	Leave open
H13	Y2	S	Y line connection	Leave open
H14	XVDD	P	High voltage power	–
H15	X4	S	X matrix drive line	Leave open
H16	X5	S	X matrix drive line	Leave open
H17	X6	S	X matrix drive line	Leave open
H18	X7	S	X matrix drive line	Leave open
H19	X8	S	X matrix drive line	Leave open
J1	DBG_DATA XOUT	O	Debug data External oscillator output	Leave open
J2	$\overline{\text{TEST}}$	–	Reserved for factory use; leave open	Leave open
J3	VDDCORE	P	Digital power	–
J4	Y35	S	Y line connection	Leave open
J5	Y32	S	Y line connection	Leave open
J6	Y28	S	Y line connection	Leave open
J7	Y24	S	Y line connection	Leave open
J8	AVDD	P	Analog power	–

Table 2-1. Pin Listing – UFBGA 162 Balls (Continued)

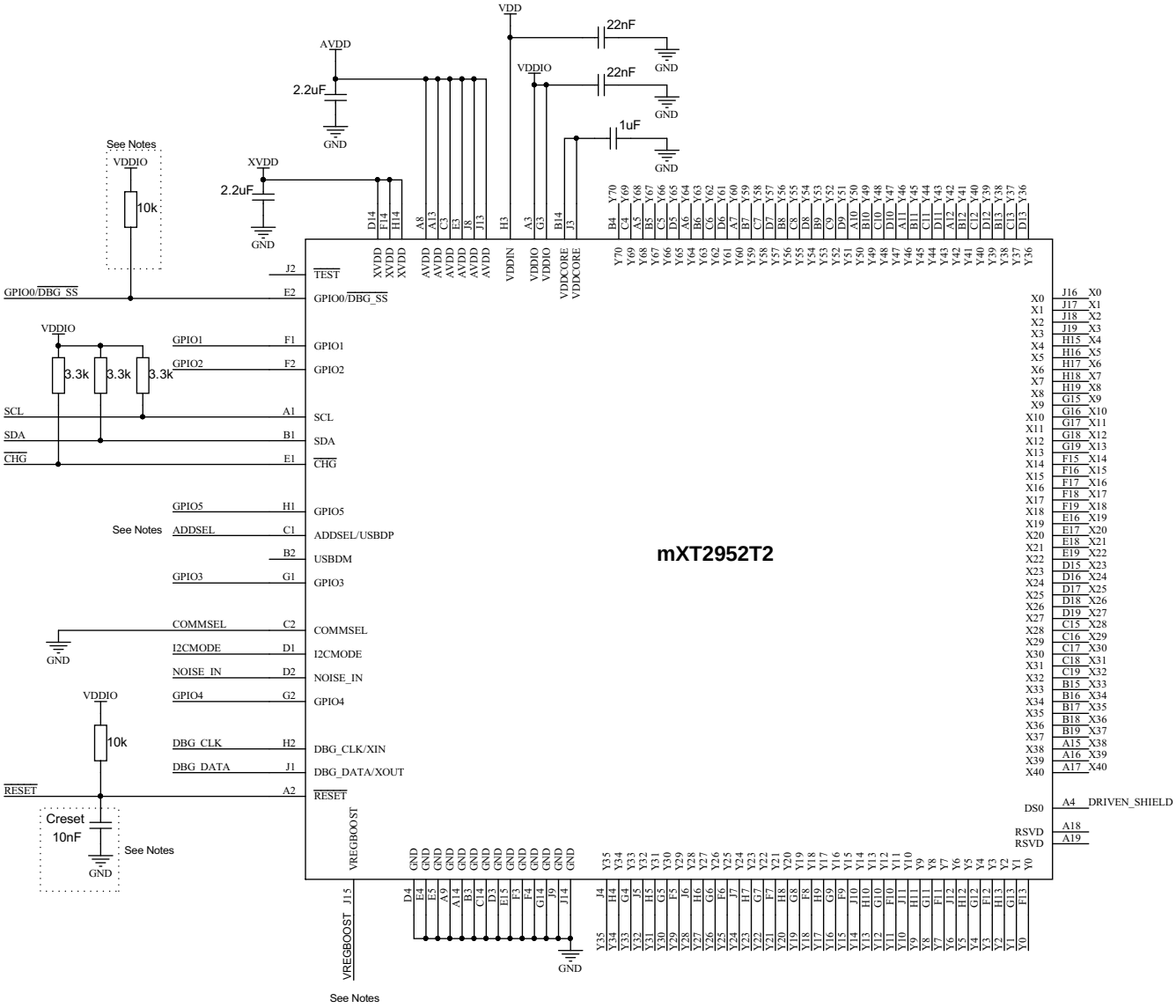
Pin	Name	Type	Description	If Unused...
J9	GND	P	Ground	–
J10	Y14	S	Y line connection	Leave open
J11	Y10	S	Y line connection	Leave open
J12	Y6	S	Y line connection	Leave open
J13	AVDD	P	Analog power	–
J14	GND	P	Ground	–
J15	VREGBOOST	O	Voltage booster control	Leave open
J16	X0	S	X matrix drive line	Leave open
J17	X1	S	X matrix drive line	Leave open
J18	X2	S	X matrix drive line	Leave open
J19	X3	S	X matrix drive line	Leave open

Key:

I	Input only	O	Output only	I/O	Input or output
OD	Open drain output	P	Ground or power	S	Sense pin

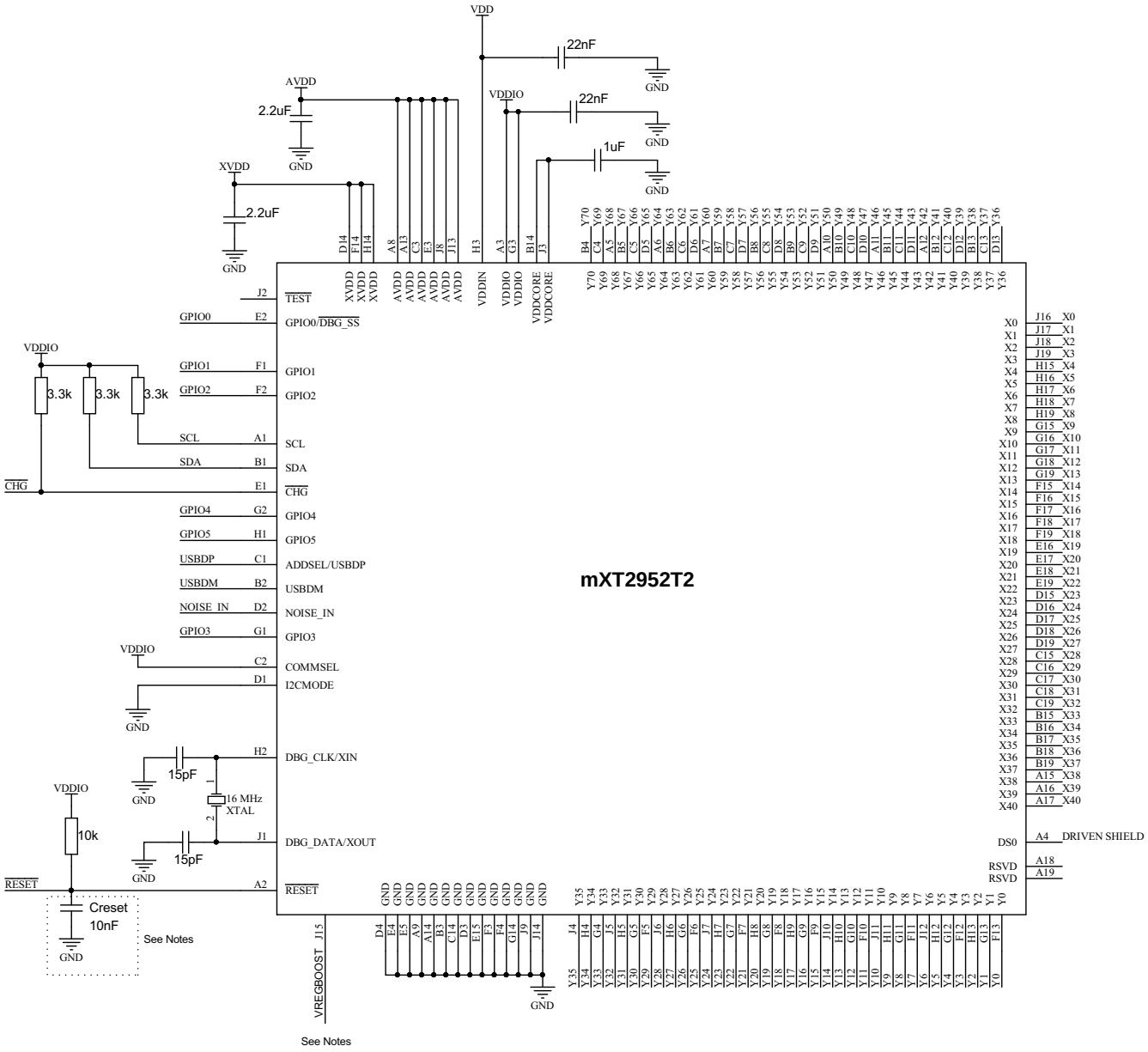
3. Schematic

3.1 UFBGA 162 Balls – I²C Mode



See “Schematic Notes” on page 16.

3.2 UFBGA 162 Balls – USB Mode



See “Schematic Notes” on page 16.

3.3 Schematic Notes

3.3.1 VDDCORE

VddCore is internally generated from the Vdd 3.3 V power supply. To guarantee stability of the internal voltage regulator, a minimum value of 1 µF must be used for decoupling on VDDCORE.

3.3.2 $\overline{\text{DBG\_SS}}$ Line

The $\overline{\text{DBG\_SS}}$ line shares the same ball as GPIO0. Only one of these two functions can be chosen and the circuit should be designed accordingly.

In I^2C mode, pull-up resistor R2 in the schematics is optional and should be present only if the ball is used as $\overline{\text{DBG\_SS}}$.

For more information refer to Application Note: *QTAN0050 Using the maXTouch Debug Port*.

3.3.3 $\overline{\text{RESET}}$ Line

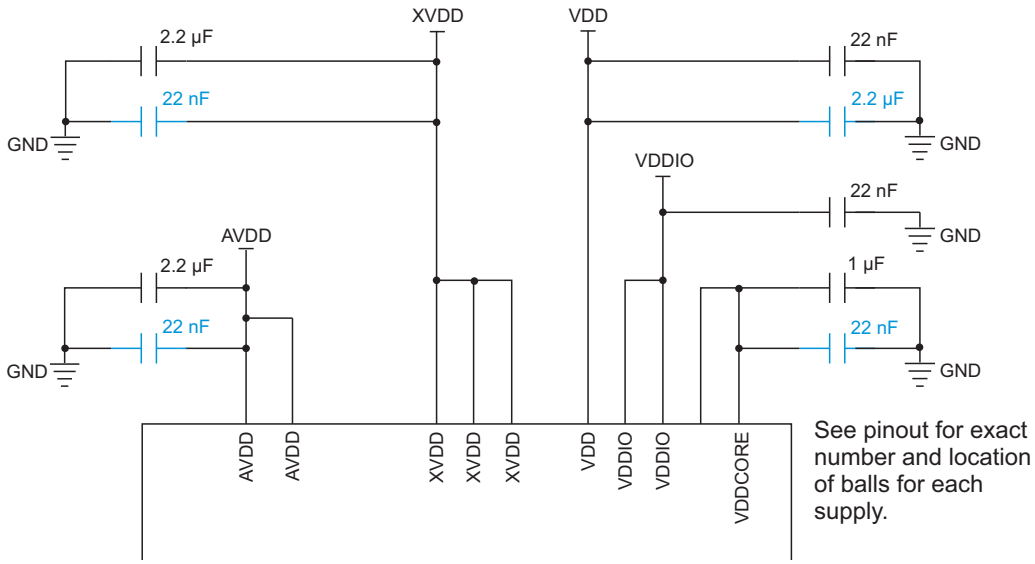
The $\overline{\text{RESET}}$ line is shown on the schematics with a 10 nF capacitor to ground. This capacitor is optional but may help if ESD issues are encountered.

3.3.4 Decoupling capacitors

1. All decoupling capacitors must be X7R or X5R and placed <5 mm away from the balls for which they act as bypass capacitors. Pins of the same type can share a capacitor provided no pin is more than 10 mm from the capacitor.
2. The schematics on the previous pages show the minimum capacitors required if the device is placed on the system board. If the ball configuration means that sharing a bypass capacitor is not possible (distance between balls too great to satisfy condition 1 or routing difficulty), then the number of base capacitors should be increased.

   Note that this requires that the voltage regulator supplies for AVdd, Vdd and VddIO are clean and noise free. It also assumes that the track length between the capacitors and on-board power supplies is < 50 mm.
3. If an active tail design is used, the voltage regulators are likely to be some distance from the device and it may be necessary to implement additional decoupling. In this case, a parallel combination of capacitors is recommended to give high and low frequency filtering as shown in Figure 3-1.

Figure 3-1. Additional Recommended Decoupling Capacitors

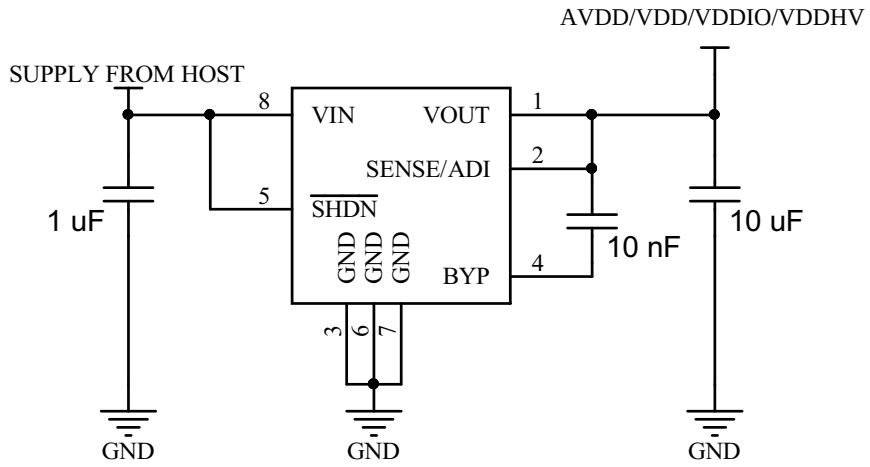


NOTES: 1. Additional base capacitors may be required if balls are widely separated.
2. Recommended additional decoupling capacitors are shown in blue

3.3.5 Low Drop-Out Voltage Regulators (LDOs)

In applications where the VddIO supply is at the same voltage level as Vdd and AVdd (that is, 3.3 V) it is permissible to use a single LDO for all supply rails (AVDD/VDD/VDDIO/VDDHV). A suitable circuit is shown in Figure 3-2.

Figure 3-2. Low Drop-Out Regulators



Where poor or inadequate tracking or decoupling leads to high noise levels on the supply rails, Atmel recommends that a separate low drop-out voltage regulator supply is used for the AVdd supply.

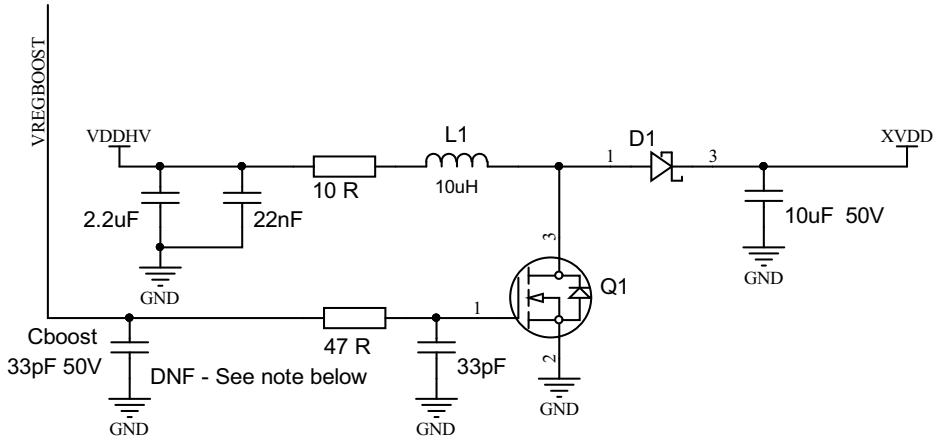
See Section 4.5 on page 19 for further details. A list of approved regulators is given in Table 4-1 on page 20.

3.3.6 Voltage Booster

The XVdd power can be supplied using the Voltage Booster shown in Figure 3-3 or an external regulated supply. See Section 16.2 on page 67 for the supply voltages possible and refer to the *mXT2952T2 1.1 Protocol Guide* for information on how to set the required voltage. To run the High Voltage boost controller in a low frequency mode, a larger inductor will need to be fitted to the boost circuit.

If an external supply is used, the components in Figure 3-3 can be omitted and VREGBOOST should be left open circuit.

Figure 3-3. XVdd Supply Circuit



Notes:
1. Do not fit capacitor Cboost but make provision for it next to the VREGBOOST ball. This capacitor may be required to minimize RF noise issues.
2. See Section 3.3.6.1 for suggested suppliers for L1 and Q1.

3.3.6.1 Suggested Component Suppliers

D1 is a Schottky Diode. Possible suppliers are shown in Table 3-1.

Table 3-1. Suitable Schottky Diode (D1)

Manufacturer	Device
Various	BAT54M3TG5
Various	1N4148WX

L1 is a 10 µH inductor in a 1812 case. Possible suppliers are shown in Table 3-2.

Table 3-2. Suitable 10 µH Inductors (L1)

Manufacturer	Device
Panasonic	ELJFB100JF
TDK	MLZ1608M100WT
TDK	MLZ2012M100WT000

Q1 is an N-channel 20 V, 700 mA, MOSFET. Possible suppliers are shown in Table 3-3.

Table 3-3. Suitable MOSFETs (Q1)

Manufacturer	Device
ON Semiconductor	NTA4153NT1G
Toshiba	SSM3K56FS
NXP	PMR290UNE (Note: NXP do not recommend use in new designs)

4. Circuit Components

4.1 Decoupling Capacitors

Each power supply pin requires decoupling as described in Section 3.3 on page 16. The capacitors should be ceramic X7R or X5R.

The PCB traces connecting the decoupling capacitors to the pins of the device must not exceed 10 mm in length. This limits any stray inductance that would reduce filtering effectiveness.

4.2 I^2C Line Pull-up Resistors

The values for pull-up resistors on SDA and SCL need to be chosen to ensure rise times are within I^2C specification – if the rise time is too long the overall clock rate will be reduced.

If using a VddIO at the low end of the allowable range it is likely that the pull-up resistor values will need to be reduced from those shown on the schematic.

4.3 Supply Quality

While the device has good Power Supply Rejection Ratio properties, poorly regulated and/or noisy power supplies can significantly reduce performance.

Always operate the device with a well-regulated and clean AVdd and XVdd supply. It supplies the sensitive analog stages in the device.

4.4 Oscillator

A 16 MHz crystal oscillator must be connected to the device when the device is operating in USB mode. A crystal oscillator with a minimum accuracy of 100 ppm must be used.

An external oscillator is not needed in I^2C mode.

4.5 Suggested Voltage Regulators

An LDO regulator should be chosen that provides adequate output capability, low noise, good load regulation and step response.

Suitable fixed output LDO devices are shown in Table 4-1 on page 20.

With a single regulator, PCB layout is more critical than with multiple LDO regulators, and special care with the PCB layout should be taken. See Section 13.5 on page 61 for information concerning PCB design with a single LDO.

4.5.1 Multiple Voltage Regulator Supply

The AVdd supply stability is critical for the device because this supply interacts directly with the analog front end. If noise problems exist when using a single LDO regulator, Atmel recommends that the supply for the analog section of the board be supplied by a regulator that is separate from the logic supply and high voltage regulators. This reduces the amount of noise injected into the sensitive, low signal level parts of the design.

4.5.2 Suggested Voltage Regulators

The voltage regulators listed in Table 4-1 have been tested and found to work well with the mXT2952T2.

Table 4-1. Suitable LDO Regulators

Manufacturer	Device	Current Rating (mA)
Analog Devices	ADP122/ADP123	300
Diodes Inc.	AP2125	300
Diodes Inc.	AP7335	300
Linear Technology	LT1763CS8-3.3	500
NXP	LD6836	300
Texas Instruments	LP2981	100
Texas Instruments	LP3981	300
Texas Instruments	LP5996	150 / 300

1. Some manufacturers claim that minimal or no capacitance is required for correct regulator operation. However, in all cases, a minimum of a 1.0 µF ceramic, low ESR capacitor at the input and output of these devices should be used. The manufacturer's datasheets should always be referred to when selecting capacitors for these devices and the typical recommended values, types and dielectrics adhered to.
2. A"soft-start" regulator with excellent noise and load step regulation will be needed to satisfy the XVdd supply requirements. 1% resistors should be used to define the nominal output voltage. If 5% resistors are used, the nominal XVdd voltage must be reduced accordingly to ensure that the recommended voltage range is adhered to.

4.5.3 LDO Selection Criteria

The LDO devices in Table 4-1 have been proved to provide satisfactory performance in Atmel maxTouch controllers, however, if it is desired to use an alternative LDO, certain performance criteria should be verified before using the device. These are:

- Stable with low value multi-layer ceramic capacitors on input and output – actual values will be device dependent, but it is good design practice to use values greater than the minimum specified in the LDO regulator data sheet
- Low output noise – less than 100 µV RMS over the range 10 Hz to 1 MHz
- Good load transient response – this should be less than 35 mV peak when a load step change of 100 mA is applied at the device output terminal
- Input supply requirement of between 4.5 V and 5.5 V
- Low quiescent current to improve battery life
- Thermal and current limit overload protection
- Ideally, select an LDO with common footprint, to allow interchanging between regulators

5. Touchscreen Basics

5.1 Sensor Construction

A touchscreen is usually constructed from a number of transparent electrodes. These are typically on a glass or plastic substrate. They can also be made using non-transparent electrodes, such as copper or carbon. Electrodes are constructed from Indium Tin Oxide (ITO) or metal mesh. Thicker electrodes yield lower levels of resistance (perhaps tens to hundreds of Ω/square) at the expense of reduced optical clarity. Lower levels of resistance are generally more compatible with capacitive sensing. Thinner electrodes lead to higher levels of resistance (perhaps hundreds to thousands of Ω/square) with some of the best optical characteristics.

Interconnecting tracks can cause problems. The excessive RC time constants formed between the resistance of the track and the capacitance of the electrode to ground can inhibit the capacitive sensing function. In such cases, the tracks should be replaced by screen printed conductive inks (non-transparent) outside the touchscreen viewing area.

5.2 Electrode Configuration

The specific electrode designs used in Atmel touchscreens are the subject of various patents and patent applications. Further information is available on request.

The device supports various configurations of electrodes as summarized in Section 6. on page 23.

5.3 Scanning Sequence

All nodes are scanned in sequence by the device. There is a full parallelism in the scanning sequence to improve overall response time. The nodes are scanned by measuring capacitive changes at the intersections formed between the first X line and all the Y lines. Then the intersections between the next X line and all the Y lines are scanned, and so on, until all X and Y combinations have been measured.

The device can be configured in various ways. It is possible to disable some nodes so that they are not scanned at all. This can be used to improve overall scanning time.

5.4 Touchscreen Sensitivity

5.4.1 Adjustment

Sensitivity of touchscreens can vary across the extents of the electrode pattern due to natural differences in the parasitic capacitance of the interconnections, control chip, and so on. An important factor in the uniformity of sensitivity is the electrode design itself. It is a natural consequence of a touchscreen pattern that the edges form a discontinuity and hence tend to have a different sensitivity. The electrodes at the far edges do not have a neighboring electrode on one side and this affects the electric field distribution in that region.

A sensitivity adjustment is available for the whole touchscreen. This adjustment is a basic algorithmic threshold that defines when a node is considered to have enough signal change to qualify as being in detect.

5.4.2 Mechanical Stackup

The mechanical stackup refers to the arrangement of material layers that exist above and below a touchscreen. The arrangement of the touchscreen in relation to other parts of the mechanical stackup has an effect on the overall sensitivity of the screen. QMatrix technology has an excellent ability to operate in the presence of ground planes close to the sensor. QMatrix sensitivity is attributed more to the interaction of the electric fields between the transmitting (X) and receiving (Y) electrodes than to the surface area of these electrodes. For this reason, stray capacitance on the X or Y electrodes does not strongly reduce sensitivity.

Front panel dielectric material has a direct bearing on sensitivity. Plastic front panels are usually suitable up to about 1.2 mm, and glass up to about 2.5 mm (dependent upon the screen size and layout). The thicker the front panel, the lower the signal-to-noise ratio of the measured capacitive changes and hence the lower the resolution of the touchscreen. In general, glass front panels are near optimal because they conduct electric fields almost twice as easily as plastic panels.

Note: Care should be taken using ultra-thin glass panels as retransmission effects can occur, which can significantly degrade performance.

6. Sensor Layout

6.1 Mutual Capacitance Matrix

The specific electrode designs used in Atmel touchscreens are the subject of various patents and patent applications. Further information is available on request.

The physical matrix can be configured to have one or more touch objects. These are configured using the appropriate touch objects (Multiple Touch Touchscreen T100 and Key Array T15). It is not mandatory to have all the allowable touch objects present. The objects are disabled by default so only those that you wish to use need to be enabled. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on configuring the touch objects.

The device supports various configurations of electrodes as summarized below:

- Touchscreen: 41 X × 71 Y maximum (subject to other configurations)
- Keys: Up to 64 keys in an X/Y grid

When designing the physical layout of the touch panel, obey the following rules:

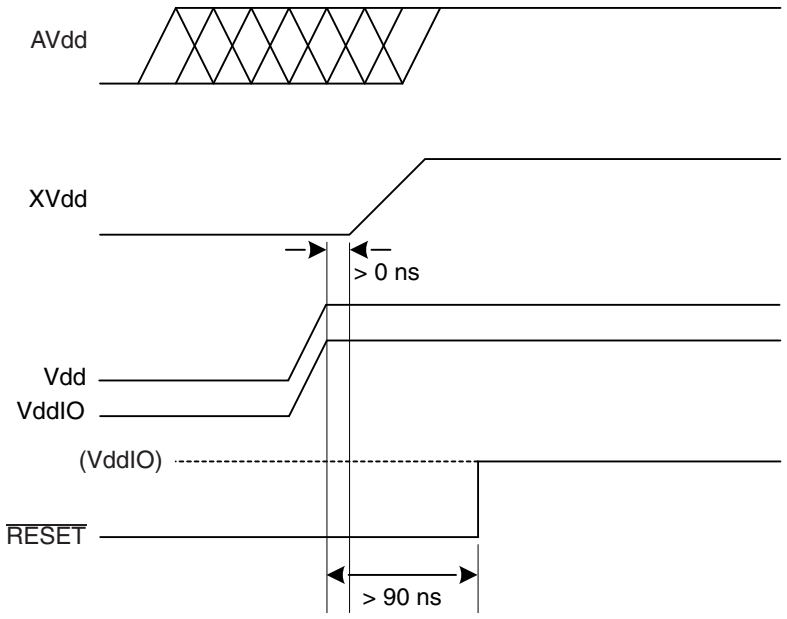
- Each touch object should be a regular rectangular shape in terms of the lines it uses
- The touch objects cannot share X and Y lines if self-capacitance measurement is enabled.
- It is recommended that the touchscreen should start at X0, Y0; if self-capacitance measurement is enabled, the touchscreen **must** start at X0, Y0
- It is recommended that the keys should occupy the highest X and Y lines

7. Power-up / Reset Requirements

There is an internal Power-on Reset (POR) in the device.

If an external reset is to be used the device must be held in RESET (active low) while the digital (Vdd) analog (AVdd) and I/O (VddIO) power supplies are powering up. The supplies must have reached their nominal values before the RESET signal is deasserted (that is, goes high). This is shown in Figure 7-1. See Section 16.2 on page 67 for nominal values for Vdd, VddIO, AVdd, and XVdd.

Figure 7-1. Power Sequencing on the mXT2952T2



Note: 1) Vdd, VddIO, and AVdd can be powered up in any order
2) XVdd must not be powered up until after Vdd and must obey the rate-of-rise specification

CAUTION: XVdd must not be grounded when Vdd is active as damage to the device may result.

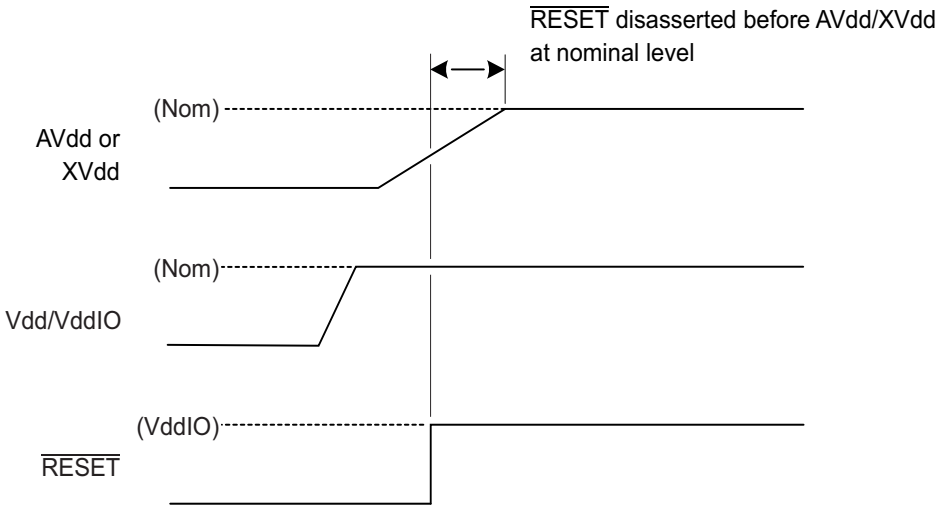
Vdd must be applied to the device before the external XVdd supply to ensure that the different power domains in the device are initialized correctly. Typically this can be done by connecting the enable pin of the Switched-Mode Power Supply (SMPS) supplying XVdd to a 10 kΩ pull-up resistor connected to the Vdd, but the XVdd can be controlled separately by the host, if required.

After power-up, the device takes 100 ms before it is ready to start communications.

If the RESET line is released before the AVdd or external XVDD supply has reached its nominal voltage (see Figure 7-2 on page 25), then some additional operations need to be carried out by the host. There are two options open to the host controller:

- Start the part in deep sleep mode and then send the command sequence to set the cycle time to wake the part and allow it to run normally. Note that in this case a calibration command is also needed.
- Send a reset command.

Figure 7-2. Power Sequencing on the mXT2952T2 – Late rise on AVDD or XVdd



The RESET pin can be used to reset the device whenever necessary. The RESET pin must be asserted low for at least 90 ns to cause a reset. After releasing the RESET pin the device takes 100 ms before it is ready to start communications. It is recommended to connect the RESET pin to a host controller to allow it to initiate a full hardware reset without requiring a power-down.

Make sure that any lines connected to the device are below or equal to Vdd during power-up. For example, if RESET is supplied from a different power domain to the VDDIO pin, make sure that it is held low when Vdd is off. If this is not done, the RESET signal could parasitically couple power via the RESET pin into the Vdd supply.

Note that the voltage level on the RESET pin of the device must never exceed VddIO (digital supply voltage).

A software reset command can be used to reset the chip (refer to the Command Processor T6 object in the *mXT2952T2 1.1 Protocol Guide*. A software reset takes a maximum of 101 ms. After the chip has finished it asserts the CHG line to signal to the host that a message is available. The reset flag is set in the Message Processor object to indicate to the host that it has just completed a reset cycle. This bit can be used by the host to detect any unexpected brownout events. This allows the host to take any necessary corrective actions, such as reconfiguration.

A checksum check is performed on the configuration settings held in the nonvolatile memory. If the checksum does not match a stored copy of the last checksum, then this indicates that the settings have become corrupted. This is signaled to the host by setting the configuration error bit in the message data for the Command Processor T6 object (refer to the *mXT2952T2 1.1 Protocol Guide* for more information).

Note that the CHG line is briefly set as an input during power-up or reset. It is therefore particularly important that the line should be allowed to float high via the CHG line pull-up resistor during this period. It should not be driven by the host (see Table 16.8.3 on page 79).

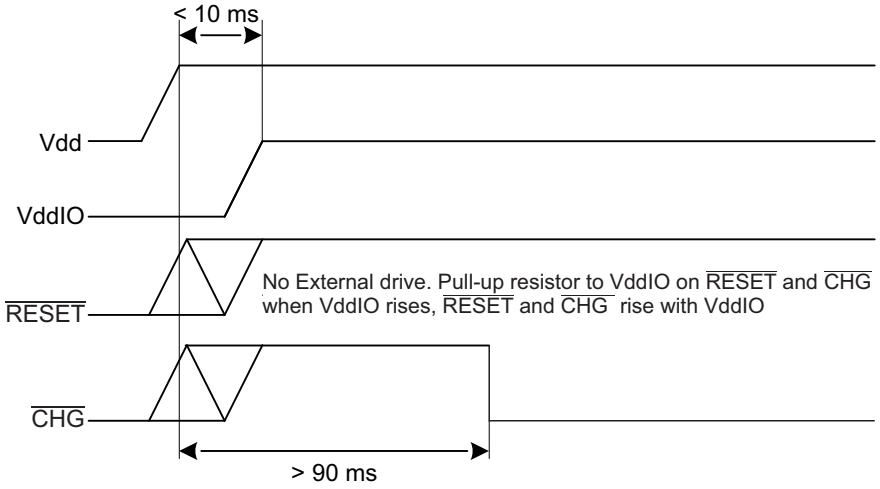
At power-on, the device performs a self-test routine to check for shorts that might cause damage to the device. Refer to the Self Test T25 object in the *mXT2952T2 1.1 Protocol Guide* for more details about this process.

7.1 Power-up and Reset Sequence – VddIO Enabled after Vdd

The Power-up sequence that can be used in applications where VddIO must be powered up after Vdd, is shown in Figure 7-3.

In this case the communication interface to mXT is not driven by the host system. The RESET and CHG pins are connected to VddIO using suitable pull-up resistors. Vdd is powered up, followed by VddIO, no more than 10 ms after Vdd. Due to the pull-up resistors, RESET and CHG will rise with VddIO. The internal POR system ensures reliable boot up of the device and the CHG line will go low approximately 90 ms after Vdd to notify the host that the device is ready to start communication.

Figure 7-3. Power-up Sequence



7.1.1 Summary

The Power-up and RESET requirements for the maXTouch devices are summarised in the table below.

Condition	External RESET	VddIO Delay (After Vdd)	AVdd Power-Up	Comments
1	Low at Power-up	0 ms	Before RESET is released	If AVdd bring-up is delayed then additional actions will be required by the host. See notes in Figure 7-1 on page 24
2	Not driven	<10 ms	Before VddIO	

8. Detailed Operation

8.1 Touch Detection

The mXT2952T2 allows for both mutual and self capacitance measurements, with the self capacitance measurements being used to augment the mutual capacitance measurements to produce reliable touch information.

When self capacitance measurements are enabled, touch classification is achieved using both mutual and self capacitance touch data. This has the advantage that both types of measurement systems can work together to detect touches under a wide variety of circumstances.

Mutual capacitance touch data is used wherever possible to classify touches as this has greater granularity than self capacitance measurements and provides positional information on touches. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on measurements.

Self capacitance measurements, on the other hand, allow for the detection of single touches in extreme case, such as single thick glove touches, when touches can only be detected by self capacitance data and may be missed by mutual capacitance touch detection.

8.2 Operational Modes

The device operates in two modes: Active (touch detected) and Idle (no touches detected). Both modes operate as a series of burst cycles. Each cycle consists of a short burst (during which measurements are taken) followed by an inactive sleep period. The difference between these modes is the length of the cycles. Those in idle mode typically have longer sleep periods. The cycle length is configured using the IDLEACQINT and ACTVACQINT settings in the Power Configuration T7. In addition, an *Active to Idle Timeout* setting is provided.

Refer to the *mXT2952T2 1.1 Protocol Guide* for full information on how these modes operate, and how to use the settings provided.

8.3 Detection Integrator

The device features a touch detection integration mechanism. This acts to confirm a detection in a robust fashion. A counter is incremented each time a touch has exceeded its threshold and has remained above the threshold for the current acquisition. When this counter reaches a preset limit the sensor is finally declared to be touched. If, on any acquisition, the signal is not seen to exceed the threshold level, the counter is cleared and the process has to start from the beginning.

The detection integrator is configured using the appropriate touch objects (Multiple Touch Touchscreen T100, Key Array T15). Refer to the *mXT2952T2 1.1 Protocol Guide* for more information.

8.4 Sensor Acquisition

The maximum acquisition time for one X line on the mXT2952T2 is 5 µs. Care should be taken to ensure that the total time for one X line configured by the Acquisition Configuration T8 and CTE Configuration T46 objects do not exceed this (refer to the *mXT2952T2 1.1 Protocol Guide* for details on these objects).

8.5 Calibration

Calibration is the process by which a sensor chip assesses the background capacitance on each node. Nodes are only calibrated on reset and when:

- The node is enabled (that is, activated).

or

- The node is already enabled and one of the following applies:
  - The node is held in detect for longer than the Touch Automatic Calibration setting (refer to the *mXT2952T2 1.1 Protocol Guide* for more information on TCHAUTOCAL setting in the Acquisition Configuration object).

- The signal delta on a node is at least the touch threshold (TCHTHR) in the anti-touch direction, while it meets the criteria in the Touch Recovery Processes that results in a recalibration. (Refer to the *mXT2952T2 1.1 Protocol Guide* for objects Acquisition Configuration T8 and Self Capacitance Configuration T111).
- The host issues a recalibrate command.
- Certain configuration settings are changed.

A status message is generated on the start and completion of a calibration.

Note that the device performs a global calibration; that is, all the nodes are calibrated together.

8.6 Digital Filtering and Noise Suppression

The mXT2952T2 supports on-chip filtering of the acquisition data received from the sensor. Specifically, the maXCharger T72 object provides an algorithm to suppress the effects of noise (for example, from a noisy charger plugged into the user's product). This algorithm can automatically adjust some of the acquisition parameters on-the-fly to filter the analog-to-digital conversions (ADCs) received from the sensor.

Additional noise suppression is provided by the Self Capacitance maXCharger T108 object. Similar in both design and configuration to the maXCharger T72 object, the Self Capacitance maXCharger T108 object is the noise suppression interface for self capacitance touch measurements.

Noise suppression is triggered when a noise source is detected.

- A hardware trigger can be implemented using the NOISE_IN pin.
- The host driver code can indicate when a noise source is present.
- The noise suppression is also triggered based on the noise levels detected using internal line measurements.The maXCharger T72 and Self Capacitance maXCharger T108 object selects the appropriate controls to suppress the noise present in the system.

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the maXCharger T72 and Self Capacitance maXCharger T108 objects.

8.7 Shieldless Support and Display Noise Suppression

The mXT2952T2 can support shieldless sensor design even with a noisy LCD by using the following features.

- **Optimal Integration:** This feature is not filtering as such, but enables the user to use a shorter integration window. The integration window optimizes the amount of charge collected against the amount of noise collected, to ensure an optimal SNR. This feature also benefits the system in the presence of an external noise source. This feature is configured using the Shieldless T56 object. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information
- **Display noise suppression:** This feature is based on filtering provided by the Lens Bending T65 object (See Section 8.10 on page 29). This feature allows the device to overcome display noise simultaneously with external noise. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information

8.8 Retransmission Compensation

The device can limit the undesirable effects on the mutual capacitance touch signals caused by poor device coupling to ground, such as poor sensitivity and touch break-up. This is achieved using the Retransmission Compensation T80 object. This object can be configured to allow the touchscreen to compensate for signal degradation due to these undesirable effects. If self capacitance measurements are also scheduled, the Retransmission Compensation T80 object will use the resultant data to enhance the compensation process.

The Retransmission Compensation T80 object is also capable of compensating for water presence on the sensor if self capacitance measurements are scheduled. In this case, both mutual capacitance and self capacitance measurements are used to detect moisture and then, once moisture is detected, self capacitance measurements are used to detect single touches in the presence of moisture.

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the Retransmission Compensation T80 object.

8.9 Grip Suppression

The device has two grip suppression mechanisms to suppress false detections from a user's grip.

Mutual grip suppression works by specifying a boundary around a touchscreen, within which touches can be suppressed whilst still allowing touches in the center of the touchscreen. This ensures that a "rolling" hand touch (such as when a user grips a mobile device) is suppressed. A "real" (finger) touch towards the center of the screen is allowed.

Mutual grip suppression is configured using the Grip Suppression T40 object. There is one instance of the Grip Suppression T40 object for each Multiple Touch Touchscreen T100 object present on the device.

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the Grip Suppression T40 object.

Self Capacitance grip suppression works by looking for characteristic shapes in the self capacitance measurement along the touchscreen boundary, and thereby distinguishing between a grip and a touch further into the sensor.

8.10 Lens Bending

The device supports algorithms to eliminate disturbances from the measured signal and also to measure the bend component.

When the sensor suffers from the screen deformation (lens bending) the signal values acquired by normal procedure are corrupted by the disturbance component (bend). The amount of bend depends on:

- The mechanical and electrical characteristics of the sensor
- The amount and location of the force applied by the user touch to the sensor

The Lens Bending T65 object measures the bend component and compensates for any distortion caused by the bend. As the bend component is primarily influenced by the user touch force, it can be used as a secondary source to identify the presence of a touch. The additional benefit of the Lens Bending T65 object is that it will eliminate LCD noise as well.

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the Lens Bending T65 object.

8.11 Glove Detection

The device has glove detection algorithms that process the measurement data received from the touchscreen classifying touches as potential gloved touches.

The Glove Detection T78 object is used to detect glove touches. In Normal Mode the Glove Detection T78 object applies vigorous glove classification to small signal touches to minimize the effect of unintentional hovering finger reporting. Once a gloved touch is found, the Glove Detection T78 object enters Glove Confidence Mode. In this mode the device expects the user to be wearing gloves so the classification process is much less stringent.

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the Glove Detection T78 object.

8.12 Stylus Support

The mXT2952T2 allows for the particular characteristics of passive stylus touches, whilst still allowing conventional finger touches to be detected. The touch sensitivity and threshold controls for stylus touches are configured separately from those for conventional finger touches so that both types of touches can be accommodated.

Stylus support ensures that the small touch area of a stylus registers as a touch, as this would otherwise be considered too small for the touchscreen. Additionally, there are controls to distinguish a stylus touch from an unwanted approaching finger (such as on the hand holding the stylus).

Passive stylus touches are configured by the Stylus T47 object. There is one instance of the Stylus T47 object for each Multiple Touch Touchscreen T100 object present on the device.

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on configuring a stylus.

8.13 Unintentional Touch Suppression

The Touch Suppression T42 object provides a mechanism to suppress false detections from unintentional touches from a large body area, such as from a face, ear or palm. The Touch Suppression T42 object also provides Maximum Touch Suppression to suppress all touches if more than a specified number of touches has been detected. There is one instance of the Touch Suppression T42 object for each Multiple Touch Touchscreen T100 object present on the device.

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the Touch Suppression T42 object.

8.14 Adjacent Key Suppression Technology

Adjacent Key Suppression (AKS) technology is a patented method used to detect which touch object is touched when objects are located close together. A touch in a group of AKS objects is only indicated on the object in that group that is touched first. This is assumed to be the intended object. Once an object in an AKS group is in detect, there can be no further detections within that group until the object is released. Objects can be in more than one AKS group.

Note that AKS technology works best when it operates in conjunction with a detect integration setting of several acquisition cycles.

The device has two levels of AKS. The first level works between the touch objects (Multiple Touch Touchscreen T100 and Key Array T15). The touch objects are assigned to AKS groups. If a touch occurs within one of the touch objects in a group, then touches within other objects inside that group are suppressed. For example, if a touchscreen and a Key Array are placed in the same AKS group, then a touch in the touchscreen will suppress touches in the Key Array, and vice versa.

The second level of AKS is internal AKS within an individual Key Array object (note that internal AKS is not present on other types of touch objects, only a Key Array T15). If internal AKS is enabled, then when one key is touched, touches on all the other keys within the Key Array are suppressed.

AKS is configured using the touch objects (Multiple Touch Touchscreen T100 or Key Array T15).

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information.

Note: If a touch is in detect and then AKS is enabled, that touch will not be forced out of detect. It will not go out of detect until the touch is released. AKS will then operate normally. This applies to both levels of AKS.

8.15 GPIO Pins

The mXT2952T2 has 6 GPIO pins. The pins can be set to be either an input or an output, as required. Note that unused GPIO pins can be left externally unconnected as long as they are given a defined state by using the GPIO/PWM Configuration T19 object. With the GPIO/PWM Configuration T19 object, an unused GPIO pin can be either set to Input mode, with internal pull-up, or Output mode.

By default GPIO pins are set to be inputs. If not used they should be connected to GND. Alternatively, they can be set as outputs using the GPIO/PWM Configuration T19 object and left open.

9. Host Communications

9.1 Communication Mode Selection (COMMSEL Pin)

The selection of the I²C or USB interface is determined by the COMMSEL pin:

- If the COMMSEL pin is low (GND), I²C mode is selected.
- If the COMMSEL pin is high (VddIO), USB mode is selected.

9.2 I²C Mode Selection (I2CMODE Pin)

The selection of the I²C or the HID-I²C mode is determined by the I2CMODE pin:

- If the I2CMODE pin is low (GND), HID-I²C mode is selected.
- If the I2CMODE pin is high (VddIO), I²C mode is selected.
- If the I2CMODE pin is floating, the mode is selected according to the I²C address (as determined by the ADDSEL pin). See Section 9.2.1 for more information.

9.2.1 Automatic Selection of I²C and HID-I²C Modes

If the I2CMODE pin is left floating (that is, automatic mode selection), the device will listen on both I²C addresses and automatically select the protocol to be used depending on the first message received. In this case the ADDSEL pin determines the primary and secondary I²C addresses, and these in turn determine the communications mode to be used. If the primary I²C address is detected, I²C is used for communications; if the I²C secondary address is detected, HID-I²C is used.

The selection of both the communications mode and the I²C addresses is summarized in Table 9-1 on page 31.

Table 9-1. Communications Mode Selection

I2CMODE	ADDSEL	Mode
0 (HID-I²C selected)	0 (Address = 0x4A)	HID-I²C communications at 0x4A
	1 (Address = 0x4B)	HID-I²C communications at 0x4B
1 (I²C selected)	0 (Address = 0x4A)	I²C communications at 0x4A
	1 (Address = 0x4B)	I²C communications at 0x4B
No input or input floating (auto selection)	0 (Primary address = 0x4A, secondary address = 0x4B)	I²C communications at 0x4A (primary address) HID-I²C communications at 0x4B (secondary address)
	1 (Primary address = 0x4B, secondary address = 0x4A)	I²C communications at 0x4B (primary address) HID-I²C communications at 0x4A (secondary address)

9.3 I²C Address Selection (ADDSEL Pin)

If the I2CMODE pin is not floating (that is, a particular mode is chosen), the ADDSEL pin selects a single I²C address, according to Table 9-2.

Table 9-2. I²C Address Selection

ADDSEL	I²C Address
Connected to GND	0x4A
Pulled up to VddIO	0x4B

10. I²C Communications

The device can use an I²C interface for communication. See Appendix B. on page 87 for details of the I²C protocol.

The I²C interface is used in conjunction with the $\overline{CHG}$ line. The $\overline{CHG}$ line going active signifies that a new data packet is available. This provides an interrupt-style interface and allows the device to present data packets when internal changes have occurred.

To use the device in I²C mode, the I2CMODE pin should be pulled high. Alternatively, if auto selection is required, the I2CMODE pin should be left floating and the primary I²C address used. See Section 9. on page 31 for more information.

10.1 I²C Addresses

The device supports two I²C device addresses that are selected using the ADDSEL line at start up. The two internal I²C device addresses are 0x4A and 0x4B. The selection of the address (and the communication mode) is described in Section 9.3 on page 32. These are shifted left to form the SLA+W or SLA+R address when transmitted over the I²C interface, as shown in Figure 10-1.

Table 10-1. Format of an I²C Address

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Address: 0x4A or 0x4B							Read/write

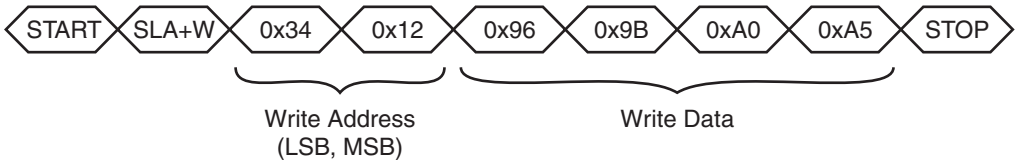
10.2 Writing To the Device

A WRITE cycle to the device consists of a START condition followed by the I²C address of the device (SLA+W). The next two bytes are the address of the location into which the writing starts. The first byte is the Least Significant Byte (LSByte) of the address, and the second byte is the Most Significant Byte (MSByte). This address is then stored as the address pointer.

Subsequent bytes in a multi-byte transfer form the actual data. These are written to the location of the address pointer, location of the address pointer + 1, location of the address pointer + 2, and so on. The address pointer returns to its starting value when the WRITE cycle STOP condition is detected.

Figure 10-1 shows an example of writing four bytes of data to contiguous addresses starting at 0x1234.

Figure 10-1. Example of a Four-byte Write Starting at Address 0x1234

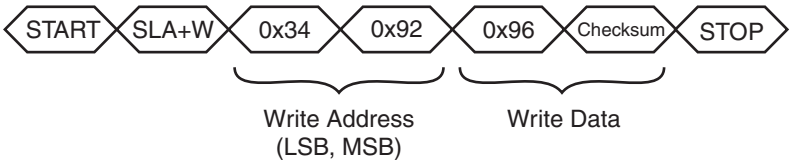


10.3 I²C Writes in Checksum Mode

In I²C checksum mode an 8-bit CRC is added to all I²C writes. The CRC is sent at the end of the data write as the last byte before the STOP condition. All the bytes sent are included in the CRC, including the two address bytes. Any command or data sent to the device is processed even if the CRC fails.

To indicate that a checksum is to be sent in the write, the most significant bit of the MSByte of the address is set to 1. For example, the I²C command shown in Figure 10-2 writes a value of 150 (0x96) to address 0x1234 with a checksum. The address is changed to 0x9234 to indicate checksum mode.

Figure 10-2. Example of a Write To Address 0x1234 With a Checksum



10.4 Reading From the Device

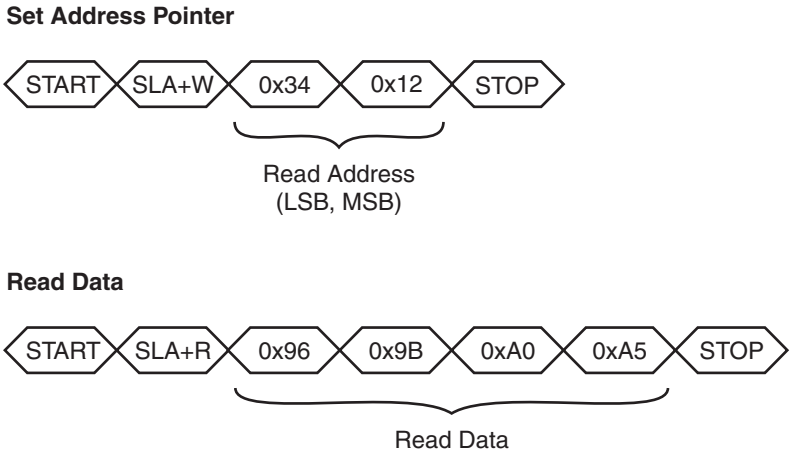
Two I²C bus activities must take place to read from the device. The first activity is an I²C write to set the address pointer (LSByte then MSByte). The second activity is the actual I²C read to receive the data. The address pointer returns to its starting value when the read cycle NACK is detected.

It is not necessary to set the address pointer before every read. The address pointer is updated automatically after every read operation. The address pointer will be correct if the reads occur in order. In particular, when reading multiple messages from the Message Processor T5 object, the address pointer is automatically reset to allow continuous reads (see Section 10.5).

The WRITE and READ cycles consist of a START condition followed by the I²C address of the device (SLA+W or SLA+R respectively).

Figure 10-3 shows the I²C commands to read four bytes starting at address 0x1234.

Figure 10-3. Example of a Four-byte Read Starting at Address 0x1234



10.5 Reading Status Messages with DMA

The device facilitates the easy reading of multiple messages using a single continuous read operation. This allows the host hardware to use a direct memory access (DMA) controller for the fast reading of messages, as follows:

1. The host uses a write operation to set the address pointer to the start of the Message Count T44 object, if necessary (1). If a checksum is required on each message, the most significant bit of the MSByte of the read address must be set to 1.
2. The host starts the read operation of the message by sending a START condition.
3. The host reads the Message Count T44 object (one byte) to retrieve a count of the pending messages (refer to the *mXT2952T2 1.1 Protocol Guide* for details).
4. The host calculates the number of bytes to read by multiplying the message count by the size of the Message Processor T5 object (2).

1. The STOP condition at the end of the read resets the address pointer to its initial location, so it may already be pointing at the Message Count T44 object following a previous message read.
2. The host should have already read the size of the Message Processor T5 object in its initialization code.

5. Note that the size of the Message Processor T5 object as recorded in the Object Table includes a checksum byte. If a checksum has not been requested, one byte should be deducted from the size of the object. That is: number of bytes = count × (size – 1).
6. The host reads the calculated number of message bytes. It is important that the host does *not* send a STOP condition during the message reads, as this will terminate the continuous read operation and reset the address pointer. No START and STOP conditions must be sent between the messages.
7. The host sends a STOP condition at the end of the read operation after the last message has been read. The NACK condition immediately before the STOP condition resets the address pointer to the start of Message Count T44 object.

Figure 10-4 shows an example of using a continuous read operation to read three messages from the device without a checksum. Figure 10-5 on page 36 shows the same example with a checksum.

Figure 10-4. Continuous Message Read Example – No Checksum

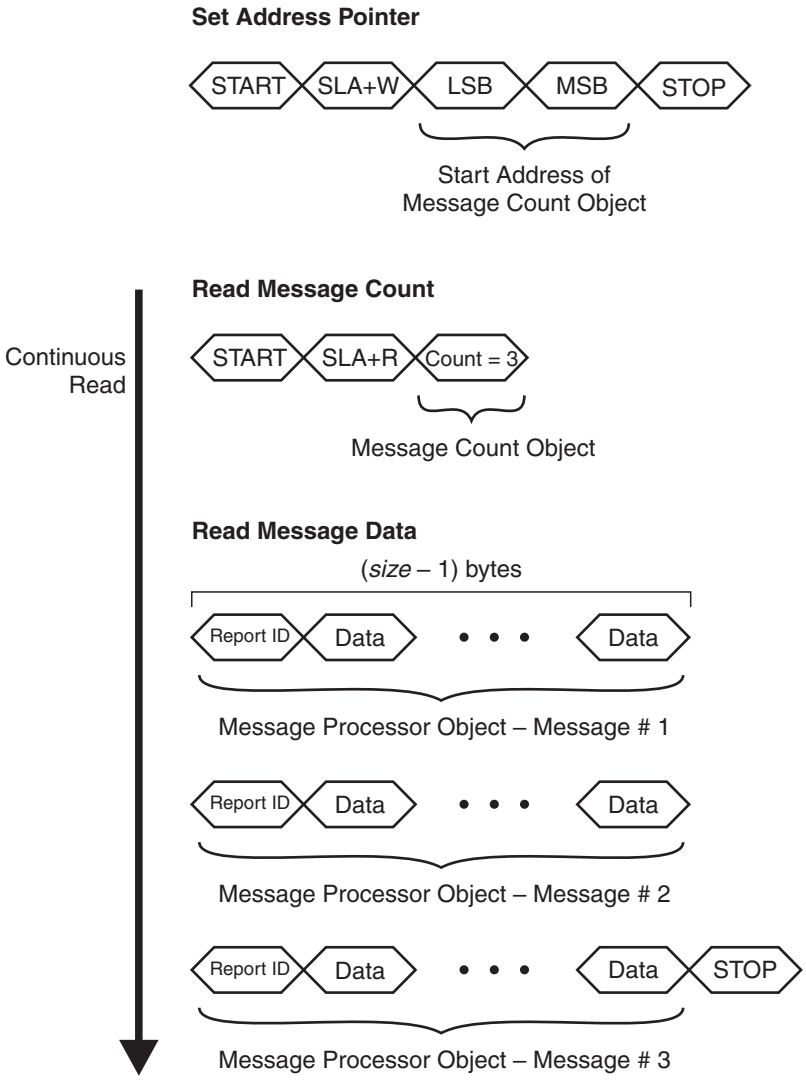
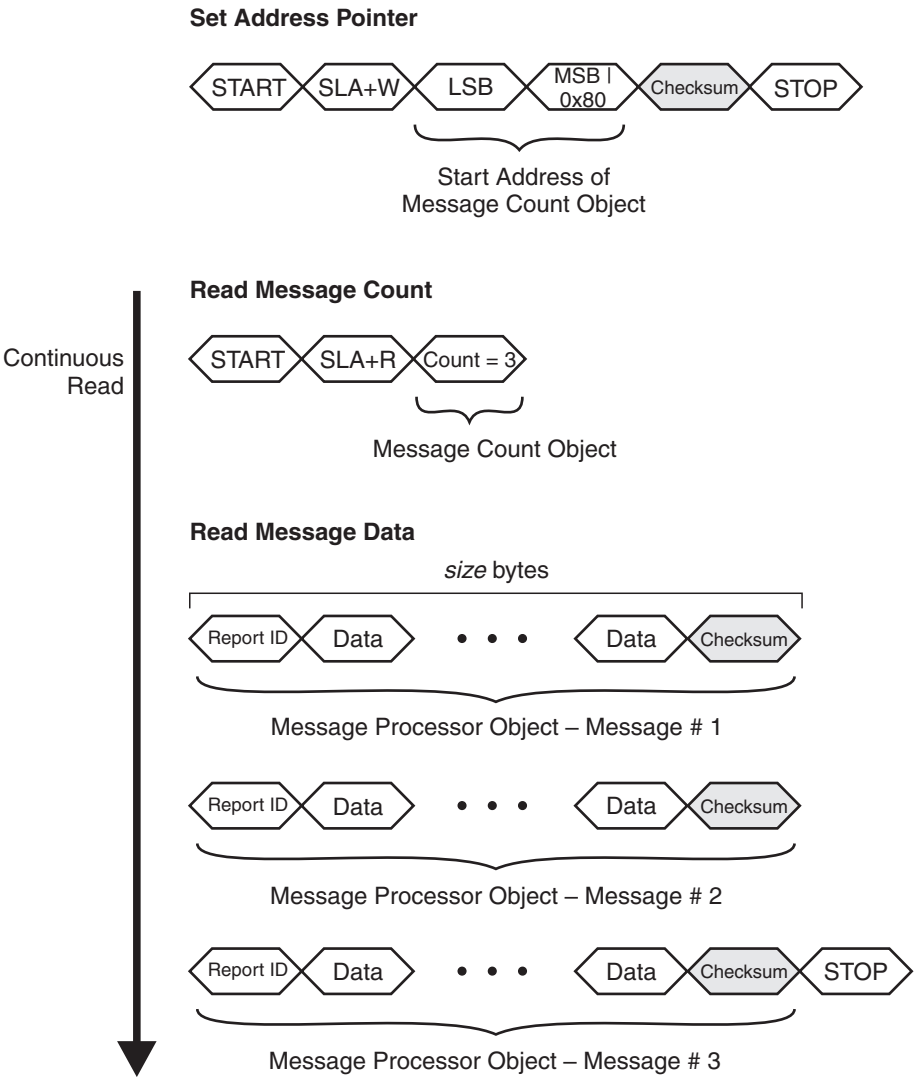


Figure 10-5. Continuous Message Read Example – I²C Checksum Mode



There are no checksums added on any other I²C reads. An 8-bit CRC can be added, however, to all I²C writes, as described in Section 10.3 on page 33.

An alternative method of reading messages using the $\overline{\text{CHG}}$ line is given in Section 10.6.

10.6 $\overline{\text{CHG}}$ Line

The $\overline{\text{CHG}}$ line is an active-low, open-drain output that is used to alert the host that a new message is available in the Message Processor T5 object. This provides the host with an interrupt-style interface with the potential for fast response times. It reduces the need for wasteful I²C communications.

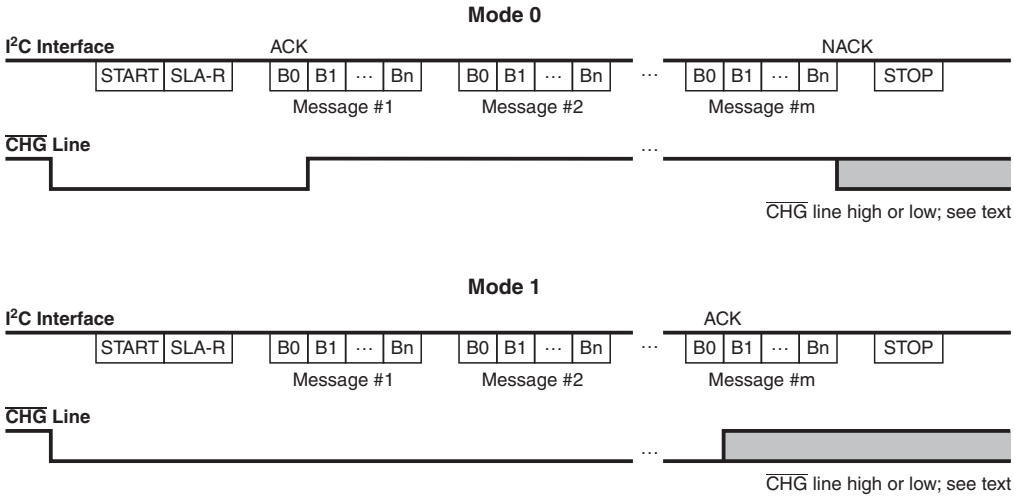
The $\overline{\text{CHG}}$ line remains low as long as there are messages to be read. The host should be configured so that the $\overline{\text{CHG}}$ line is connected to an interrupt line that is level-triggered. The host should not use an edge-triggered interrupt as this means adding extra software precautions.

The $\overline{\text{CHG}}$ line should be allowed to float during normal usage. This is particularly important after power-up or reset (see Section 7. on page 24).

A pull-up resistor is required, typically 3.3 kΩ to VddIO.

The $\overline{\text{CHG}}$ line operates in two modes, as defined by the Communications Configuration T18 object (refer to the *mXT2952T2 1.1 Protocol Guide*).

Figure 10-6. $\overline{\text{CHG}}$ Line Modes for I²C-compatible Transfers



In Mode 0:

1. The $\overline{\text{CHG}}$ line goes low to indicate that a message is present.
2. The $\overline{\text{CHG}}$ line goes high when the first byte of the first message (that is, its report ID) has been sent and acknowledged (ACK sent) and the next byte has been prepared in the buffer.
3. The STOP condition at the end of an I²C transfer causes the $\overline{\text{CHG}}$ line to stay high if there are no more messages. Otherwise the $\overline{\text{CHG}}$ line goes low to indicate a further message.

Mode 0 allows the host to continually read messages. Messaging reading ends when a report ID of 255 (“invalid message”) is received. Alternatively the host ends the transfer by sending a NACK after receiving the last byte of a message, followed by a STOP condition. If and when there is another message present, the $\overline{\text{CHG}}$ line goes low, as in step 1. In this mode the state of the $\overline{\text{CHG}}$ line does not need to be checked during the I²C read.

In Mode 1:

1. The $\overline{\text{CHG}}$ line goes low to indicate that a message is present.
2. The $\overline{\text{CHG}}$ line remains low while there are further messages to be sent after the current message.
3. The $\overline{\text{CHG}}$ line goes high again only once the first byte of the last message (that is, its report ID) has been sent and acknowledged (ACK sent) and the next byte has been prepared in the output buffer.

Mode 1 allows the host to continually read the messages until the $\overline{\text{CHG}}$ line goes high, and the state of the $\overline{\text{CHG}}$ line determines whether or not the host should continue receiving messages from the device.

Note: The state of the $\overline{\text{CHG}}$ line should be checked only between messages and not between the bytes of a message. The precise point at which the $\overline{\text{CHG}}$ line changes state cannot be predicted and so the state of the $\overline{\text{CHG}}$ line cannot be guaranteed between bytes.

The Communications Configuration T18 object can be used to configure the behavior of the $\overline{\text{CHG}}$ line. In addition to the $\overline{\text{CHG}}$ line operation modes described above, this object allows the use of edge-based interrupts, as well as direct control over the state of the $\overline{\text{CHG}}$ line. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information.

10.7 SDA, SCL

The I²C bus transmits data and clock with SDA and SCL, respectively. These are open-drain. The device can only drive these lines low or leave them open. The termination resistors (Rp) pull the line up to Vdd if no I²C device is pulling it down.

The termination resistors should be chosen so that the rise times on SDA and SCL meet the I²C specifications for the interface speed being used, bearing in mind other loads on the bus (see Section 16.10 on page 79).

10.8 Clock Stretching

The device supports clock stretching in accordance with the I^2C specification. It may also instigate a clock stretch if a communications event happens during a period when the device is busy internally. The maximum clock stretch is approximately 10 – 15 ms.

The device has an internal bus monitor that can reset the internal I^2C hardware if SDA or SCL is stuck low for more than 200 ms. This means that if a prolonged clock stretch of more than 200 ms is seen by the device, then any ongoing transfers with the device may be corrupted. The bus monitor is enabled or disabled using the Communications Configuration T18 object. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information.

11. HID-I²C Communications

The device is an HID-I²C device presenting two Top-level Collections (TLCs):

- **Generic HID-I²C** – Provides a generic HID-I²C interface that allows the host to communicate with the device using the object-based protocol (OBP).
- **Digitizer HID-I²C** – Supplies touch information to the host. This interface is supported by Microsoft Windows 8 and 8.1 without the need for additional software.

See Section 9. on page 31 for information on selecting HID-I²C mode.

Other features are identical to standard I²C communication described in Section 10. on page 33.

Refer to the Microsoft HID-I²C documentation, *HID Over I²C Protocol Specification – Device Side*, for information on the HID-I²C specification.

11.1 I²C Addresses

See Section 10.1 on page 33.

11.2 Device

The device is compliant with HID-I²C specification V1.0. It has the following specification:

Vendor ID:	0x03EB (Atmel)
Product ID:	0x214E (mXT2952T2)
Version:	16-bit Version & Build Identifier in the form 0xVVBB, where: VV = Version Major (Upper 4 bits) / Minor (Lower 4 bits) BB = Build number in BCD format
HID descriptor address:	0x0000

11.3 HID Descriptor

The host should read the HID descriptor on initialization to ascertain the key attribute of the HID device. These include the report description and the report ID to be used for communication with the HID device. The HID descriptor address is 0x0000.

Note that the host driver must not make any assumptions about the report packet formats, data locations or report IDs. These must be read from the HID descriptor as they may change in future versions of the firmware.

For more information on how to read the HID descriptor, refer to the Microsoft HID-I²C documentation.

11.4 HID-I²C Report IDs

Table 11-1 describes the HID-I²C report IDs used in reports sent to the host.

Table 11-1. HID-I²C Report IDs

Report ID	Description	Top-level Collection
0x06	Object Protocol (OBP) command and response (see Section 11.5 on page 40)	Generic HID-I²C
0x01	Touch report (see Section 11.6.1 on page 44)	Digitizer HID-I²C
0x02	Maximum Touches (Surface Contacts) report (see Section 11.6.3 on page 47)	Digitizer HID-I²C
0x05	Touch Hardware Quality Assurance (THQA) report (see Section 11.6.4 on page 47)	Digitizer HID-I²C

11.5 Generic HID-I²C

The Generic HID-I²C TLC supports an input report for receiving data from the device and an output report for sending data to the device.

Commands are sent by the host using the output reports. Responses from the device are sent using input reports.

Supported commands are:

- Read/Write Memory Map
- Send Auto-return Messages

The HID-I²C Report ID used is that for Object Protocol commands and responses; see Table 11-1 on page 39 for the value.

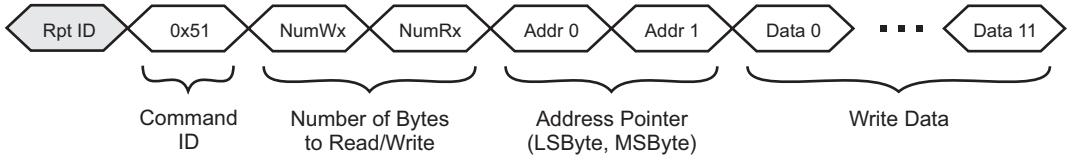
11.5.1 Read/Write Memory Map

11.5.1.1 Introduction

This command is used to carry out a write/read operation on the memory map of the device.

The command packet has the generic format given in Figure 11-1. The following sections give examples on using the command to write to the memory map and to read from the memory map.

Figure 11-1. Generic Command Packet Format

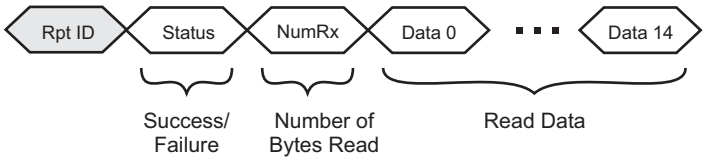


In Figure 11-1:

- **Rpt ID** is the HID-I²**C** Report ID used for Object Protocol commands and responses (see Table 11-1 on page 39).
- **NumWx** is the number of data bytes to write to the memory map (may be zero). If the address pointer is being sent, this must include the size of the address pointer.
- **NumRx** is the number of data bytes to read from the memory map (may be zero).
- **Addr 0** and **Addr 1** form the address pointer to the memory map (where necessary; may be zero if not needed).
- **Data 0** to **Data 11** are the bytes of data to be written (in the case of a write). Note that data locations beyond the number specified by NumWx will be ignored.

The response packet has the generic format given in Figure 11-2.

Figure 11-2. Response Packet Format



In Figure 11-2:

- **Rpt ID** is the HID-I²**C** Report ID used for Object Protocol commands and responses (see Table 11-1 on page 39).
- **Status** indicates the result of the command:
  - 0x00 = read and write completed; read data returned
  - 0x04 = write completed; no read data requested
- **NumRx** is the number of bytes following that have been read from the memory map (in the case of a read). This will be the same value as NumRx in the command packet.
- **Data 0** to **Data 14** are the data bytes read from the memory map.

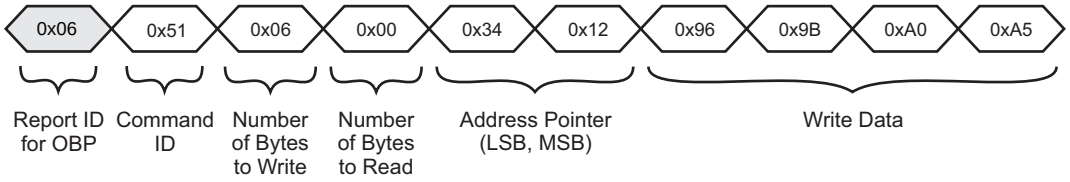
11.5.1.2 Writing To the Device

A write operation cycle to the device consists of sending a packet that contains six header bytes. These specify the HID-I^{2}C report ID, the Command ID, the number of bytes to read, the number of bytes to write, and the 16-bit address pointer.

Subsequent bytes in a multi-byte transfer form the actual data. These are written to the location of the address pointer, location of the address pointer +1, location of the address pointer + 2, and so on.

Figure 11-3 shows an example command packet to write four bytes of data to contiguous addresses starting at 0x1234.

Figure 11-3. Example of a Four-byte Write Starting at Address 0x1234

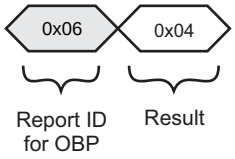


In Figure 11-3:

- **Rpt ID** is the HID-I^2**C** Report ID used for Object Protocol commands and responses (see Table 11-1 on page 39).
- The number of bytes to read is set to zero as this is a write-only operation.
- The number of bytes to write is six: that is, four data bytes plus the two address pointer bytes.

Figure 11-4 shows the response to this command. Note that the result status returned is 0x04 (that is, the write operation was completed but no read data was requested). Note also that the Report ID is the same one used in the command packet.

Figure 11-4. Response to Example Four-byte Write

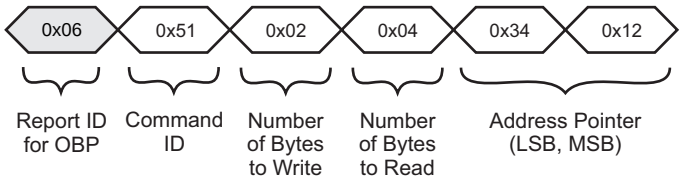


11.5.1.3 Reading From the Device

A read operation consists of sending a packet that contains the six header bytes only and no write data.

Figure 11-5 shows an example command packet to read four bytes starting at address 0x1234. Note that the address pointer is included in the number of bytes to write, so the number of bytes to write is set to 2 as there are no other data bytes to be written.

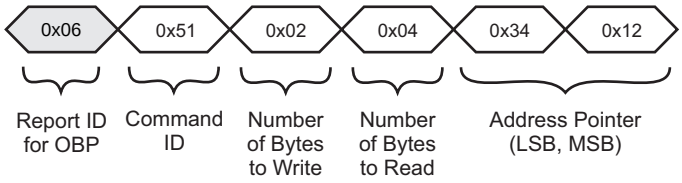
Figure 11-5. Example of a Four-byte Read Starting at Address 0x1234



It is not necessary to set the address pointer before every read. The address pointer is updated automatically after every read operation, so the address pointer will be correct if the reads occur in order.

Figure 11-6 shows the response to this command. The result status returned is 0x00 (that is the write operation was completed and the data was returned). The number of bytes returned will be the same as the number requested (4 in this case).

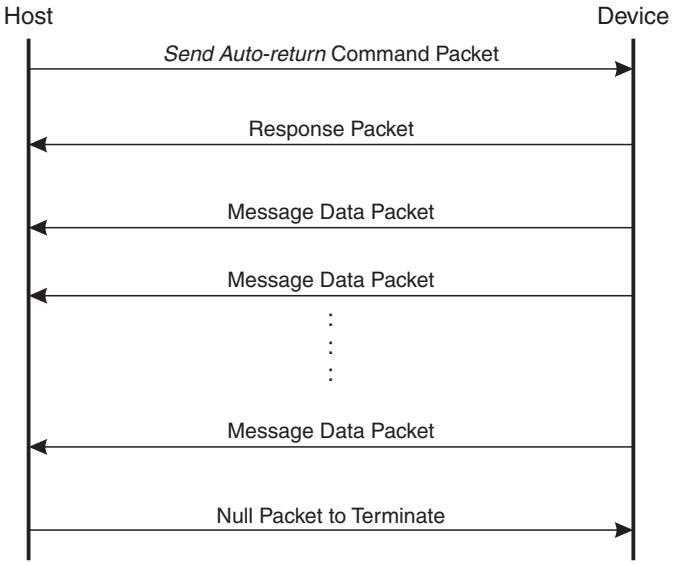
Figure 11-6. Response to Example Four-byte Read



11.5.2 Send Auto-return Messages

With this command the device can be configured to return new messages from the Message Processor T5 object autonomously. The packet sequence to do this is shown in Figure 11-7.

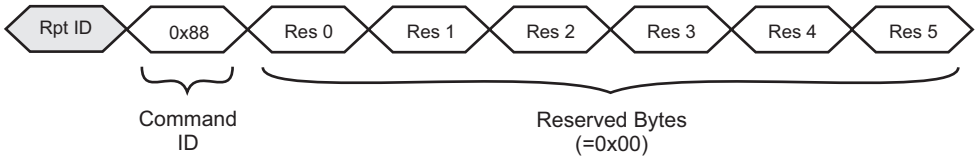
Figure 11-7. Packet Sequence for "Send Auto-return" Command



The HID-I^{2}C Report ID used is that for Object Protocol commands and responses; see Table 11-1 on page 39 for the value.

The command packet has the format given in Figure 11-8.

Figure 11-8. Command Packet Format

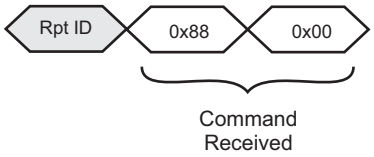


In Figure 11-8:

- **Rpt ID** is the HID-I^2**C** Report ID used for Object Protocol commands and responses (see Table 11-1 on page 39).
- **Res 0** to **Res 5** are reserved bytes with a value of 0x00.

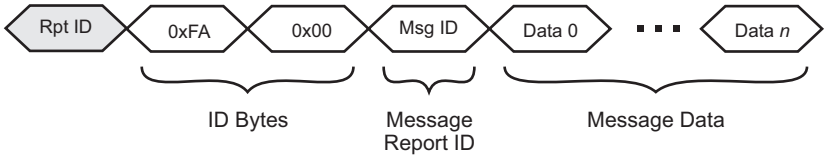
The response packet has the format given in Figure 11-9. Note that with this command, the command packet does not include an address pointer as the device already knows the address of the Message Processor T5 object.

Figure 11-9. Response Packet Format



Once the device has responded to the command, it starts sending message data. Each time a message is generated in the Message Processor T5 object, the device automatically sends a message packet to the host with the data. The message packets have the format given in Figure 11-10.

Figure 11-10. Message Packet Format

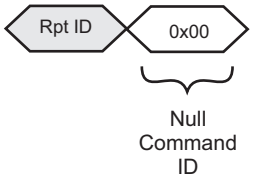


In Figure 11-10:

- **Rpt ID** is the HID-I^{2}C Report ID used for Object Protocol commands and responses (see Table 11-1 on page 39).
- **ID Bytes** identify the packet as an auto-return message packet.
- **Msg ID** is the Report ID [1] returned by the Message Processor T5 object.
- **Message Data** bytes are the bytes of data returned by the Message Processor T5 object. The size of the data depends on the source object for which this is the message data. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information.

To stop the sending of the messages, the host can send a null command packet. This consists of two bytes: a HID-I^{2}C report ID and a command byte of 0x00 (see Figure 11-11).

Figure 11-11. Null Command Packet Format



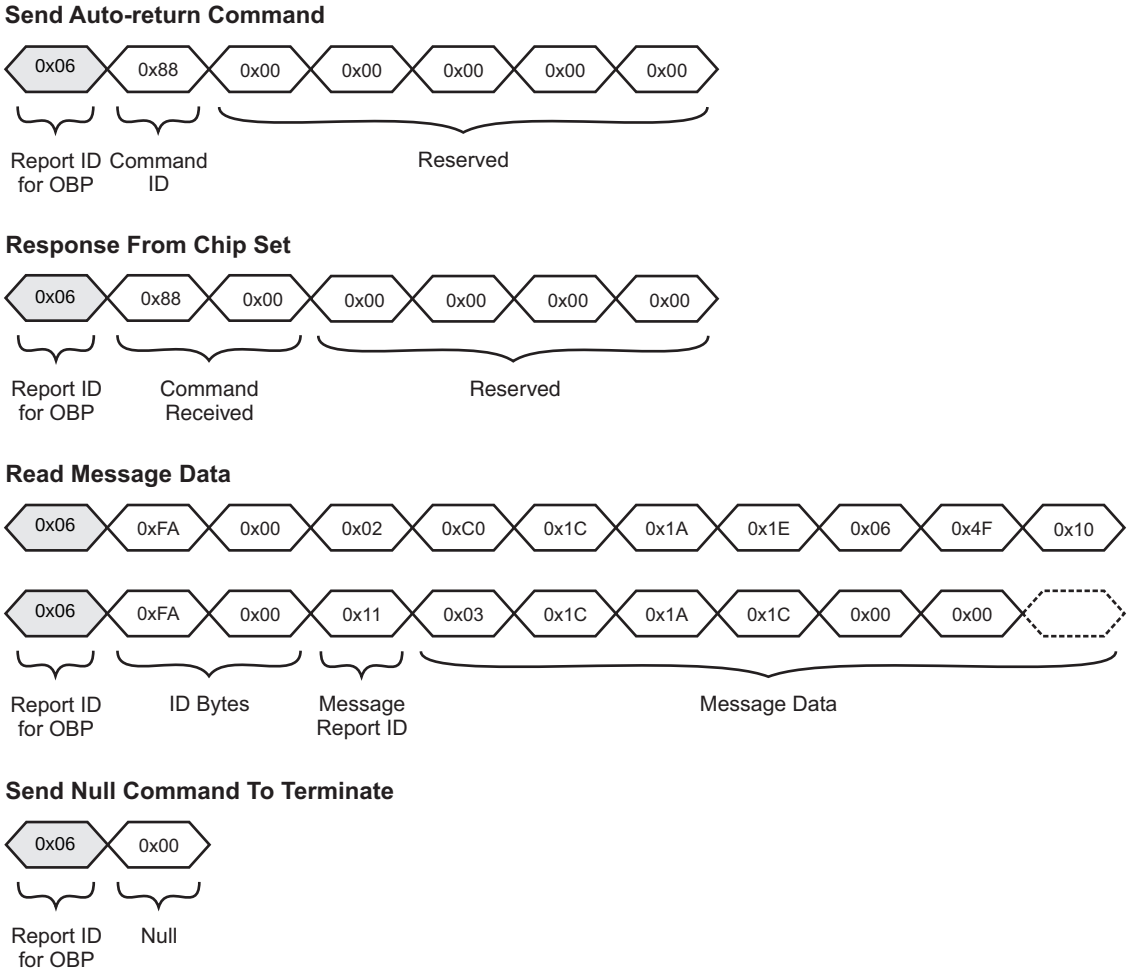
Note that any read or write will also terminate any currently enabled auto-return mode (see Section 11.5.1.2 on page 41).

1. This is the Report ID used in the Object Protocol and should not be confused with the HID-I^{2}C Report ID. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the use of Report IDs in the Object Protocol.

11.5.2.1 Reading Status Messages

Figure 11-12 shows an example sequence of packets to receive messages from the Message Processor T5 object using the “Send Auto-return” command.

Figure 11-12. Example Auto-return Command Packet



11.6 Digitizer HID-I²C

This is a digitizer class HID.

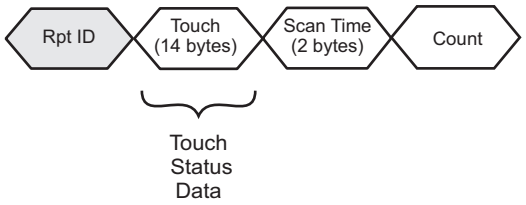
11.6.1 Touch Report

The format of a Touch report is shown in Figure 11.6.2 on page 45.

Each Touch report starts with a report ID and contains the data for one touch.

The HID-I^{2}C Report ID used is that for Touch reports; see Table 11-1 on page 39 for the value.

11.6.2 Touch Report Packet Format

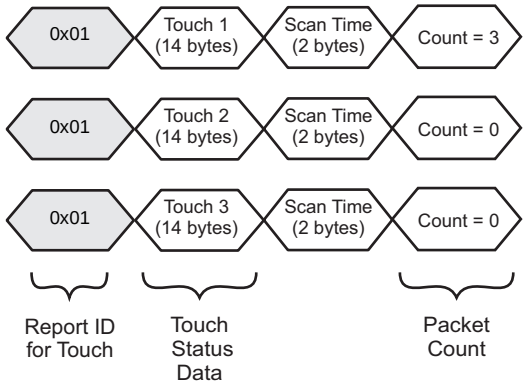


In Figure 11.6.2:

- **Rpt ID** is the HID-I^{2}C Report ID used for Touch reports (see Table 11-1 on page 39).
- **Touch** is the data for the touch.
- **Scan Time** is the Timestamp for the report packet
- **Count** is used to identify the report packets for current active touches that are to be reported as a single package. The Count in the first packet for the first touch is set to the number of active touches to be sent in one package. Subsequent packets for subsequent active touches have a Count of 0.

An example of the Touch report packets for 3 active touches is shown in Figure 11-13.

Figure 11-13. Example Touch Report Packets for 3 Active Touches



Each input report consists of a HID-I^{2}C report ID followed by 17 bytes of that describe the status of one active touch. The input report format depends on the geometry calculation control (TCHGEOMEN) of the Digitizer HID Configuration T43 object. Table 11-2 on page 46 and Table 11-3 on page 46 give the detailed format of a touch report packet.

Table 11-2. Touch Report Format when TCHGEOMEN = 1

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	HID-I²C Touch Report ID							
1	Reserved							Status
2	Touch ID							
3 – 4	X Position							
5 – 6	X' Position							
7 – 8	Y Position							
9 – 10	Y' Position							
11	Touch Width							
12	Reserved							
13	Touch Height							
14	Reserved							
15 – 16	Scan Time							
17	Count							

Table 11-3. Touch Report Format when TCHGEOMEN = 0

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	HID-I²C Touch Report ID							
1	Reserved							Status
2	Touch ID							
3 – 4	X Position							
5 – 6	X' Position							
7 – 8	Y Position							
9 – 10	Y' Position							
11	Reserved							
12	Reserved							
13	Reserved							
14	Reserved							
15 – 16	Scan Time							
17	Count							

- **Byte 0:**
  The HID-I²C Report ID (see Table 11-1 on page 39 for Touch reports).
- **Byte 1:**
  Status: Status of the touch detection. This bit is set to 1 if touch is detected, and set to 0, if no touches are detected.
- **Byte 2:**
  Touch ID: Identifies the touch for which this is a status report (starting from 0).
- **Bytes 3 to 10:**

X and Y positions: These are scaled to 12-bit resolution. This means that the upper four bits of the MSByte will always be zero.

- **Byte 11:**

  Touch Width: Reports the width of the detected touch when TCHGEOMEN is set to 1.
  Reserved when TCHGEOMEN is set to 0

- **Byte 13:**

  Touch Height: Reports the height of the detected touch when TCHGEOMEN is set to 1.
  Reserved when TCHGEOMEN is set to 0

- **Byte 15 to 16:**

  Scan Time: Timestamp associated with the current report packet with a 10 kHz resolution.

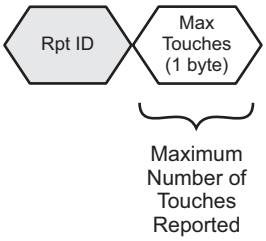
- **Byte 17:**

  Count: For the first touch, the number of active touches to be sent in one package. Subsequent packets for subsequent active touches have a Count of 0.

11.6.3 Maximum Touches (Surface Contacts) Report

The format of the Maximum Touches report packet is shown in Figure 11-14.
The HID-I^{2}C Report ID used is that for Maximum Touches reports; see Table 11-1 on page 39 for the value.

Figure 11-14. Example Maximum Touches Report



In Figure 11-14:

- **Rpt ID** is the HID-I^{2}C Report ID used for Maximum Touches reports (see Table 11-1 on page 39).
- **Max Touches** is the maximum number of touches that can be reported by the device.

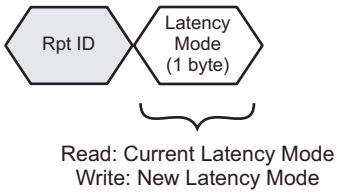
Read this report to receive the maximum number of touches that can currently be reported.

Write this report to set the maximum number of touches to be reported. Note that the number of touches cannot be set to more than the maximum number of touches defined by Multiple Touch Touchscreen T100 NUMTCH.

11.6.4 Touch Hardware Quality Assurance (THQA) Report

The THQA data is reported to Windows using the THQA Report ID (see Table 11-1 on page 39 for the value). The content of this data is defined by Microsoft.

11.7 $\overline{\text{CHG}}$ Line



The $\overline{\text{CHG}}$ line is an active-low, open-drain output that is used to alert the host that a new message is available in the Input Buffer. This provides the host with an interrupt-style interface with the potential for fast response times. It reduces the need for wasteful I^{2}C communications.

Further information on the $\overline{\text{CHG}}$ line is given in Section 10.6 on page 36.

11.8 SDA, SCL

Identical to standard I^2C operation. Refer to Section 10.7 on page 37.

11.9 Clock Stretching

Identical to standard I^2C operation. Refer to Section 10.8 on page 38.

11.10 Power Control

The mXT2952T2 supports the use of the HID-I^2C *SET POWER* commands to put the device into a low power state (analogous to the USB SUSPEND command).

11.11 Microsoft Windows Compliance

The mXT2952T2 has algorithms within the Digitizer HID Configuration T43 and Multiple Touch Touchscreen T100 specifically to ensure Microsoft Windows 8 and 8.1 compliance.

The device also supports Microsoft Touch Hardware Quality Assurance (THQA) in the Serial Data Command T68 object. Refer to the Microsoft whitepaper *How to Design and Test Multitouch Hardware Solutions for Windows 8*.

These, and other device features, may need specific tuning.

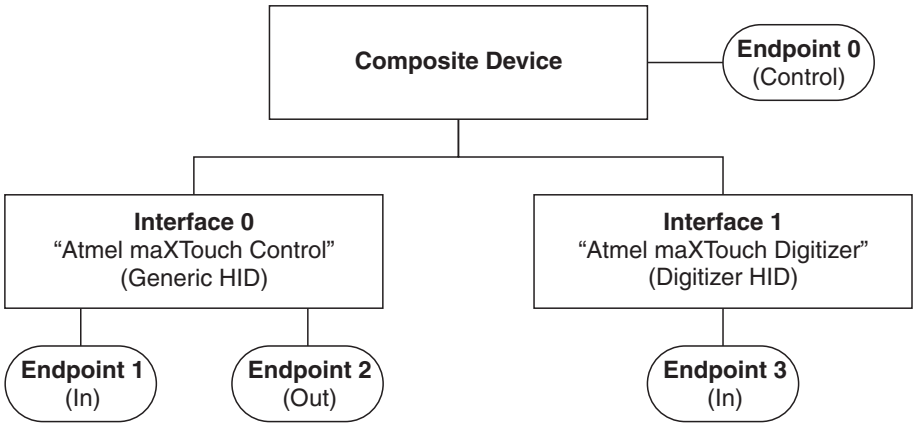
12. USB Communications

The device is a composite USB device with two Human Interface Device (HID) interfaces:

- **Interface 0** – This interface provides a Generic HID that allows the host to communicate with the device using the Object Protocol. The HID identifier string is *Atmel maXTouch Control*.
- **Interface 1** – This interface provides a Digitizer HID that supplies touch information to the Host for passing on to a PC operating system. This interface is supported by Microsoft® Windows® 8 and 8.1 without the need for additional software. The HID identifier string is *Atmel maXTouch Digitizer*.

The topography of the USB device is shown in Figure 12-1.

Figure 12-1. USB Topography



Communication takes place using Full-speed USB at 12 Mbps.

For more information on the USB HID specifications visit www.usb.org.

12.1 Endpoint Addresses

The endpoint addresses are listed in Table 12-1.

Table 12-1. Endpoint Addresses

Endpoint	Direction	Address
Endpoint 0	Bidirectional (control)	–
Endpoint 1	In	0x81
Endpoint 2	Out	0x02
Endpoint 3	In	0x83

12.2 Composite Device

The composite device is a USB 2.0-compliant USB composite device running at full speed (12 Mbps). It has the following specification:

Vendor ID: 0x03EB (Atmel)

Product ID: 0x214E (mXT2952T2)

Version: 16-bit Version & Build Identifier in the form 0xVVBB, where:
VV = Version Major (Upper 4 bits) / Minor (Lower 4 bits)
BB = Build number in BCD format

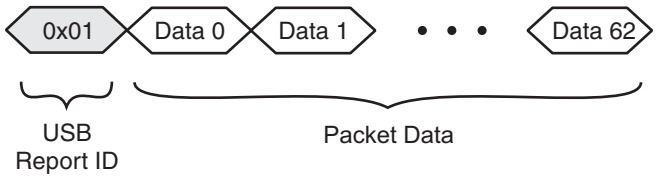
The composite device has one bidirectional endpoint: the Control Endpoint (Endpoint 0). It is used by the USB Host to interrogate the USB device for details on its configurations, interfaces and report structures. It is also used to apply general device settings relating to USB Implementation.

12.3 Interface 0 (Generic HID)

Interface 0 is a Generic Human Interface Device, compliant with HID specification 1.11 with amendments [1].

It consists of two endpoints: an interrupt-In endpoint (Endpoint 1) and an interrupt-out endpoint (Endpoint 2). The data packet in each case contains a 1-byte USB Report ID followed by 63 bytes of data, totalling 64 bytes (see Figure 12-2).

Figure 12-2. Data Packet for Interface 1



Commands are sent by the application software over the Interrupt-out endpoint, Endpoint 2. The command is sent as the first data byte of the packet data (data byte 0), followed by conditions and/or data.

The supported commands are as follows:

- Read/write Memory Map
- Send Auto-return messages
- Start debug monitoring
- End debug monitoring

Responses from the device are sent via the interrupt-In endpoint, Endpoint 1.

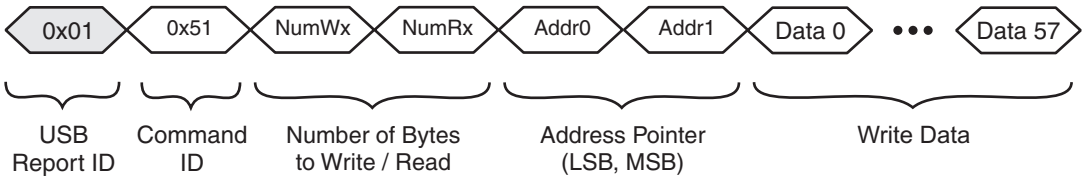
12.3.1 Read/Write Memory Map

12.3.1.1 Introduction

This command is used to carry out a write/read operation on the memory map of the device.

The USB Report ID is 0x01.The command packet has the generic format given in Figure 12-3. The following sections give examples on using the command to write to the memory map and to read from the memory map.

Figure 12-3. Generic Command Packet Form



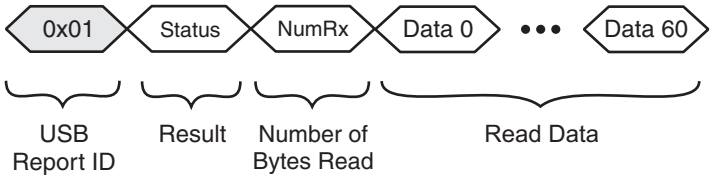
1. This is an implementation of the Microsoft USB HID specification for multi-touch digitizers.

In Figure 12-3:

- **NumWx** is the number of data bytes to write to the memory map (may be zero). If the address pointer is being sent, this must include the size of the address pointer.
- **NumRx** is the number of data bytes to read from the memory map (may be zero).
- **Addr 0** and **Addr 1** form the address pointer to the memory map (where necessary; may be zero if not needed).
- **Data 0** to **Data 57** are the bytes of data to be written (in the case of a write). Note that data locations beyond the number specified by NumWx will be ignored.

The response packet has the generic format given in Figure 12-4.

Figure 12-4. Response Packet Format



In Figure 12-4:

- **Status** indicates the result of the command:
  - 0x00 = read and write completed; read data returned
  - 0x04 = write completed; no read data requested
- **NumRx** is the number of bytes following that have been read from the memory map (in the case of a read). This will be the same value as NumRx in the command packet.
- **Data 0** to **Data 60** are the data bytes read from the memory map.

12.3.1.2 Writing To the Device

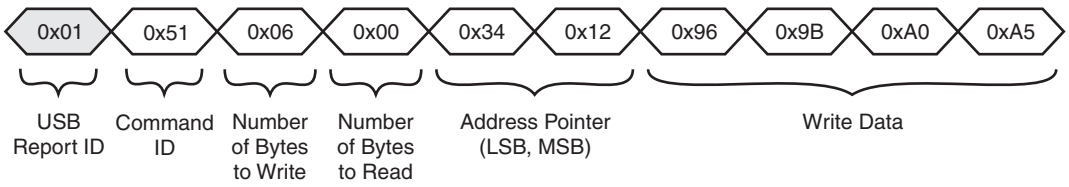
A write operation cycle to the device consists of sending a packet that contains six header bytes. These specify the USB report ID, the Command ID, the number of bytes to read, the number of bytes to write, and the 16-bit address pointer.

Subsequent bytes in a multibyte transfer form the actual data. These are written to the location of the address pointer, location of the address pointer +1, location of the address pointer + 2, and so on.

The address pointer returns to its starting value when the READ cycle STOP condition is detected.

Figure 12-5 shows an example command packet to write four bytes of data to contiguous addresses starting at 0x1234.

Figure 12-5. Example of a Four-byte Write Starting at Address 0x1234

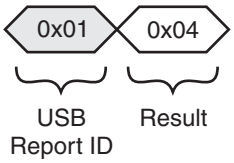


In Figure 12-5:

- The number of bytes to read is set to zero as this is a write-only operation.
- The number of bytes to write is six: that is, four data bytes plus the two address pointer bytes.

Figure 12-6 shows the response to this command. Note that the result status returned is 0x04 (that is, the write operation was completed but no read data was requested).

Figure 12-6. Response to Example Four-byte Write

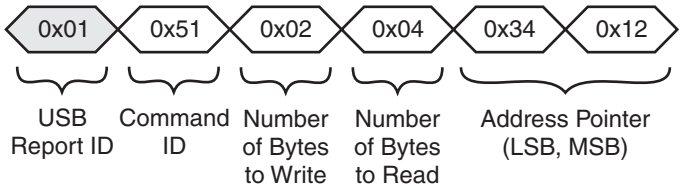


12.3.1.3 Reading From the Device

A read operation consists of sending a packet that contains the six header bytes only and no write data.

Figure 12-7 shows an example command packet to read four bytes starting at address 0x1234. Note that the address pointer is included in the number of bytes to write, so the number of bytes to write is set to 2 as there are no other data bytes to be written.

Figure 12-7. Example of a Four-byte Read Starting at Address 0x1234

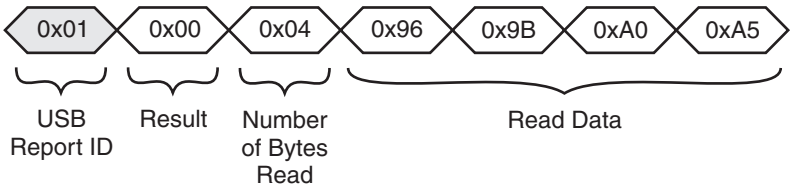


The address pointer returns to its starting value when the READ cycle STOP condition is detected.

It is not necessary to set the address pointer before every read. The address pointer is updated automatically after every read operation, so the address pointer will be correct if the reads occur in order.

Figure 12-8 shows the response to this command. The result status returned is 0x00 (that is the write operation was completed and the data was returned). The number of bytes returned will be the same as the number requested (4 in this case).

Figure 12-8. Response to Example Four-byte Read

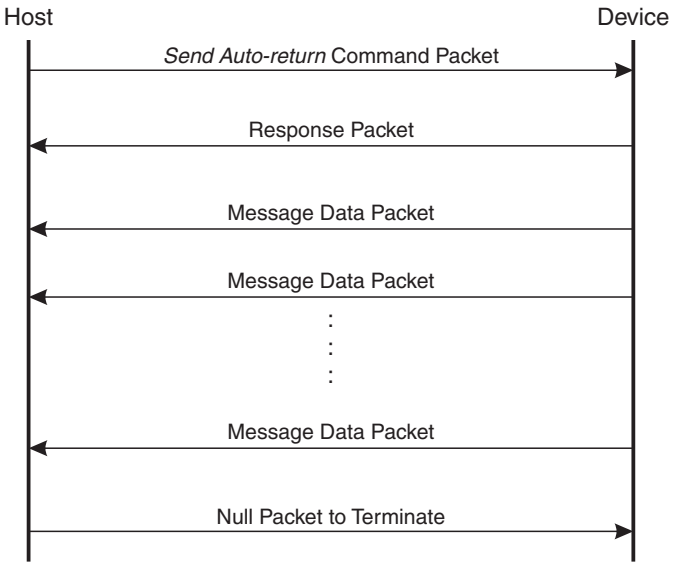


12.3.2 Send Auto-return Messages

12.3.2.1 Introduction

With this command the device can be configured to return new messages from the Message Processor T5 object autonomously. The packet sequence to do this is shown in Figure 12-9.

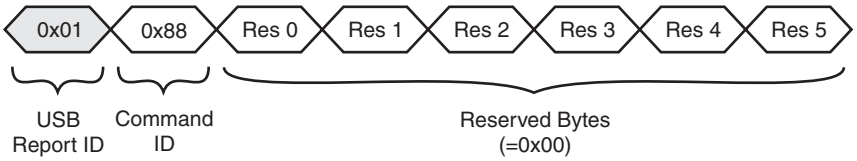
Figure 12-9. Packet Sequence for "Send Auto-return" Command



The USB Report ID is 0x01.

The command packet has the format given in Figure 12-10.

Figure 12-10. Command Packet Format

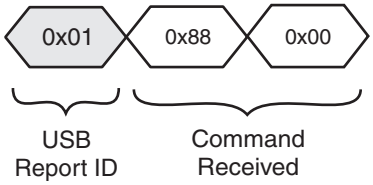


In Figure 12-10:

- **Res 0** to **Res 5** are reserved bytes with a value of 0x00.

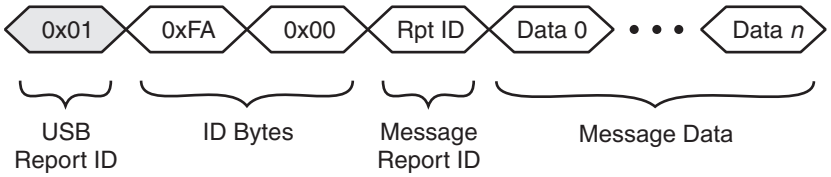
The response packet has the format given in Figure 12-11. Note that with this command, the command packet does not include an address pointer as the device already knows the address of the Message Processor T5 object.

Figure 12-11. Response Packet Format



Once the device has responded to the command, it starts sending message data. Each time a message is generated in the Message Processor T5 object, the device automatically sends a message packet to the host with the data. The message packets have the format given in Figure 12-12.

Figure 12-12.Message Packet Format

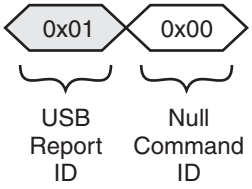


In Figure 12-12:

- **ID Bytes** identify the packet as an auto-return message packet.
- **Message Report ID** is the Report ID returned by the Message Processor T5 object. (1)
- **Message Data** bytes are the bytes of data returned by the Message Processor. The size of the data depends on the source object for which this is the message data. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information.

To stop the sending of the messages, the host can send a null command packet. This consists of two bytes: a report ID of 0x01 and a command byte of 0x00 (see Figure 12-13).

Figure 12-13.Null Command Packet Format



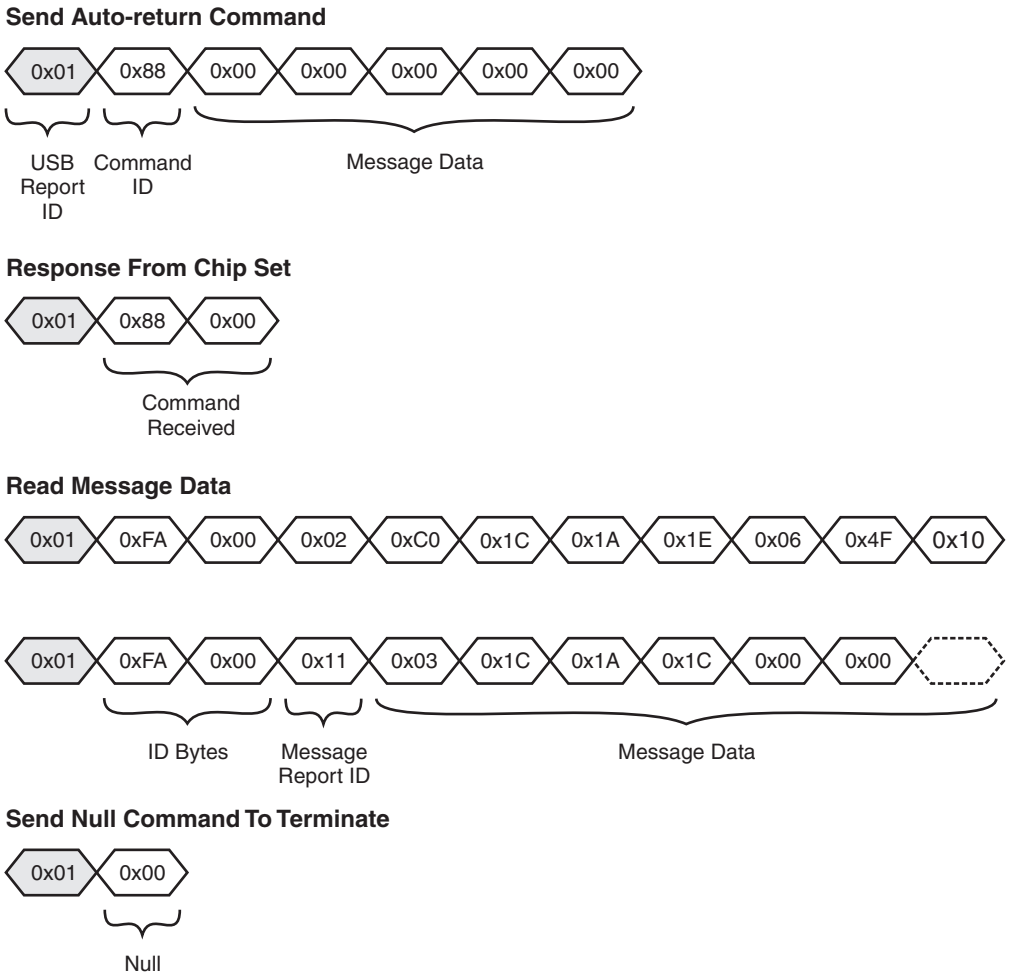
Note that the *Start Debug Monitoring* command may also terminate any currently enabled auto-return mode (see Section 12.3.2.2).

1. This is the Message Report ID used in the Object Protocol and should not be confused with the USB Report ID. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the use of Report IDs in the Object Protocol.

12.3.2.2 Reading Status Messages

Figure 12-14 on page 55shows an example sequence of packets to receive messages from the Message Processor T5 object using the “Send Auto-return” command.

Figure 12-14.Example Auto-return Command Packet



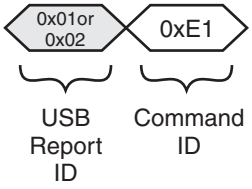
12.3.3 Start Debug Monitoring

This command instructs the device to return debug-monitoring data packets using the debug port, if this feature has been enabled in the Command Processor T6 object.

The USB Report ID can be either 0x01 or 0x02. This allows the source of the request to be identified. The main difference is that a USB Report ID of 0x01 will terminate any currently enabled auto-return mode (see Section 12.3.2 on page 52).

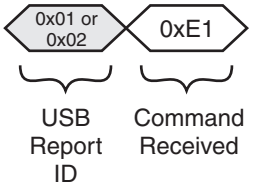
The command packet has the format given in Figure 12-15.

Figure 12-15.Command Packet Format



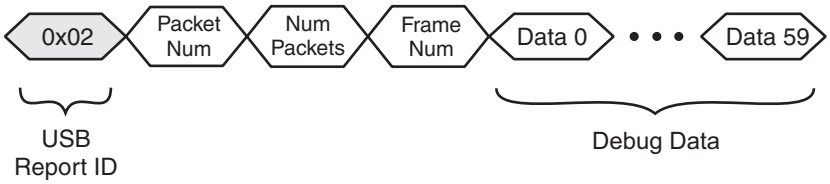
The response packet has the format given in Figure 12-16. Note that the USB Report ID will be the same as that used in the command packet.

Figure 12-16.Response Packet Format



The debug data packet has the format given in Figure 12-17.

Figure 12-17.Debug Data Packet Format



In Figure 12-17:

- **PacketNum** is the number of this USB packet in the debug data frame (full set of debug data). Refer to QTAN0050, *Using the maXTouch Debug Port*, for more information on the format of the debug data.
- **NumPackets** is the total number of USB packets that make up a debug data frame.
- **FrameNum** is the ID number of this frame.
- **Data 0** to **Data 59** are 59 bytes of debug data.

12.3.4 Stop Debug monitoring

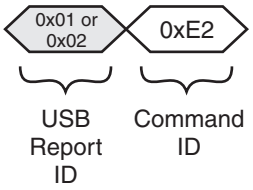
This command instructs the device to cease returning debug-monitoring data packets.

The command packet has the following format:

The USB Report ID is either 0x01 or 0x02.

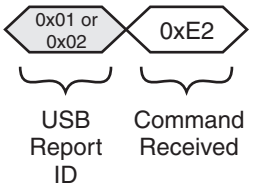
The command packet has the format given in Figure 12-18.

Figure 12-18.Command Packet Format



The response packet has the format given in Figure 12-19.

Figure 12-19.Response Packet Format



12.4 Interface 1 (Digitizer HID)

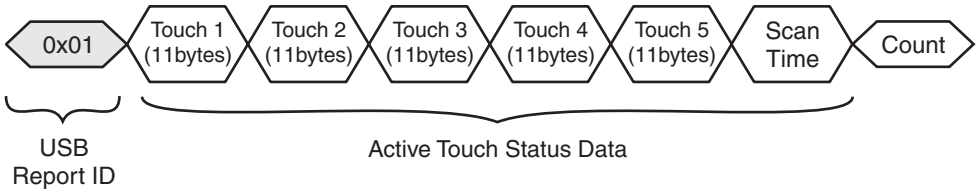
12.4.1 Normal Touch Report

Interface 1 is a Digitizer-class HID, compliant with HID specification 1.11 with amendments. [1]

This interface consists of a single interrupt-In endpoint (Endpoint 3).

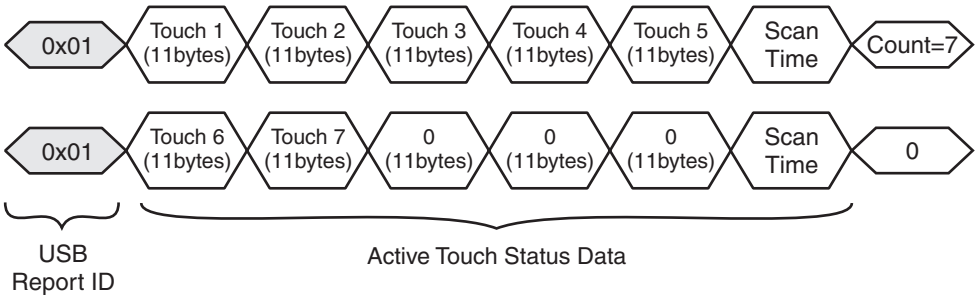
The format of an input report is shown in Figure 12-19. Each input report starts with a USB Report ID [2] (value 0x01). This is followed by 6 sets of data (11 bytes each) that describe the status of up to 5 active touches. The input report is terminated by a single byte that contains the number of active touches.

Figure 12-20.Input Report Packet



Any unused touch data bytes are set to zero (for example, the data for one active touch would be followed by 56 zeroed bytes). If there are more than five active touches to be reported, a further input report is sent with the remaining touch data. In this case, the count (for all touches) is sent in the last count byte and the count byte in the first report is zero. An example of the input report packets for 7 active touches is shown in Figure 12-21 on page 57.

Figure 12-21.Example Input Report Packets for 7 Active Touches



The input report format depends on the geometry calculation field (TCHGEOMEN) of the Digitizer HID Configuration T43 object. Table 12-2 and Table 12-3 gives the detailed format of an input report packet.

1. This is an implementation of Microsoft's USB HID specification for Multitouch digitizers.
2. The term USB Report ID should not be confused with the term Report ID as used in the Object Protocol; the two are entirely different concepts.

Table 12-2. Input Report Format when TCHGEOMEN is Enabled

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	USB Report ID							
1	Touch ID (first touch)					Reserved		Status
2	X Position LSByte (first touch)							
3	0	0	0	0	X Position MSBits (first touch)			
4	X Position LSByte (first touch)							
5	0	0	0	0	X Position MSBits (first touch)			
6	Y Position LSByte (first touch)							
7	0	0	0	0	Y Position MSBits (first touch)			
8	Y Position LSByte (first touch)							
9	0	0	0	0	Y Position MSBits (first touch)			
10	Touch width							
11	Touch height							
12 – 22	Touch data for second touch – same format as bytes 1 – 11							
23 – 33	Touch data for third touch – same format as bytes 1 – 11							
34 – 44	Touch data for fourth touch – same format as bytes 1 – 11							
45 – 55	Touch data for fifth touch – same format as bytes 1 – 11							
56 – 57	Scan time							
58	Contact count							

Table 12-3. Input Report Format when TCHGEOMEN is Disabled

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	USB Report ID							
1	Touch ID (first touch)					Reserved		Status
2	X Position LSByte (first touch)							
3	0	0	0	0	X Position MSBits (first touch)			
4 – 5	Reserved							
6	Y Position LSByte (first touch)							
7	0	0	0	0	Y Position MSBits (first touch)			
8 – 9	Reserved							
10 –11	Reserved							
12 – 22	Touch data for second touch – same format as bytes 1 – 11							
23 – 33	Touch data for third touch – same format as bytes 1 – 11							
34 – 44	Touch data for fourth touch – same format as bytes 1 – 11							
45 – 55	Touch data for fifth touch – same format as bytes 1 – 11							
56 – 57	Scan time							
58	Contact count							

In Table 12-3:

- **Byte 1:**

  Status: 1 = In detect, 0 = Not in detect.

  Touch ID: Identifies the touch for which this is a status report (starting from 1).
- **Bytes 2 to 9:**

  X and Y positions: These are scaled to 12-bit resolution. This means that the upper four bits of the MSByte will always be zero.

  Bytes 4, 5, 8, 9, 10 and 11 are Reserved when TCHGEOMEN field is set to 0.
- **Byte 10:**

  Touch Width: Reports the width of the detected touch.
- **Byte 11:**

  Touch Height: Reports the height of the detected touch.
- **Byte 56 to 57:**

  Scan Time: Timestamp associated with the current report frame with a 10 kHz resolution.

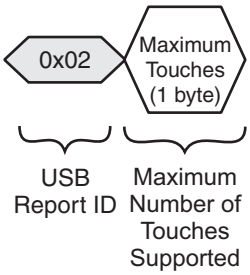
  Note that the scan time for each report packet of a single frame is same.
- **Byte 58:**

  Contact Count: Non-zero value in the first report packet of a frame indicating the total number of report packets in the frame. Zeros in the subsequent report packets within the frame.

12.4.2 Maximum Touches Report

An example of the Maximum Touches report packet is shown in Figure 12-22.

Figure 12-22.Example Maximum Touches Report



Read this report to receive the current maximum number of touches that can be reported.

12.4.3 Touch Hardware Quality Assurance (THQA) Report

The THQA data is reported to Windows using report ID 5. The content of this data is defined by Microsoft.

12.5 USB Suspend Mode

When the device is used in USB configuration, the USB “System Suspend” event can be used to minimize current consumption. Note that it is possible to put the device into deep sleep mode without also sending a “System Suspend” event on the USB bus, but the current consumption is not as low. The USB controller must send a USB “System Wake Up” event on the bus to bring the device out of suspend mode.

The device can also be configured to respond to USB “Remote Wakeup” requests. In this case, if the operating system enables remote wakeup and the device is suspended, the device will continue to scan at a preset sensor refresh rate. Use of the remote wake up feature and the sensor refresh rate are configured using the Digitizer HID Configuration T43 object (refer to the *mXT2952T2 1.1 Protocol Guide* for more information).

13. PCB Design Considerations

13.1 Introduction

The following sections give the design considerations that should be adhered to when designing a PCB layout for use with the mXT2952T2. Of these, power supply and ground tracking considerations are the most critical.

By observing the following design rules, and with careful preparation for the PCB layout exercise, designers will be assured of a far better chance of success and a correctly functioning product.

13.2 Printed Circuit Board

Atmel recommends the use of a four-layer printed circuit board for mXT2952T2 applications. This, together with careful layout, will ensure that the board meets relevant EMC requirements for both noise radiation and susceptibility, as laid down by the various national and international standards agencies.

13.2.1 PCB Cleanliness

Modern no-clean-flux is generally compatible with capacitive sensing circuits.

CAUTION: If a PCB is reworked to correct soldering faults relating to any of the device devices, or to any associated traces or components, be sure that you fully understand the nature of the flux used during the rework process. Leakage currents from hygroscopic ionic residues can stop capacitive sensors from functioning. If you have any doubts, a thorough cleaning after rework may be the only safe option.

13.3 Supply Rails and Ground Tracking

Power supply and clock distribution are the most critical parts of any board layout. Because of this, it is advisable that these be completed before any other tracking is undertaken. After these, supply decoupling, and analog and high speed digital signals should be addressed. Track widths for all signals, especially power rails should be kept as wide as possible in order to reduce inductance.

The Power and Ground planes themselves can form a useful capacitor. Flood filling for either or both of these supply rails, therefore, should be used where possible. It is important to ensure that there are no floating copper areas remaining on the board: all such areas should be connected to the 0 V plane. The flood filling should be done on the outside layers of the board.

In applications where the USB bus supplies power to the board, care should be taken to ensure that suitable capacitive decoupling is provided close to the USB connector. The tracking to the on-board regulators should also be kept as short as possible.

It should also be remembered that the screen of the USB cable is not intended to be connected to the ground or 0V supply of a remote device. It should either be left open circuit (being connected only at the host computer end) or decoupled with a suitable high voltage capacitor (typically 4.7 nF, 250 V) and a parallel resistor (typically 1 MΩ). Note that these components may not be required when the USB cabling is internal and permanently wired, and is routed away from the noisier parts of the system.

13.4 Power Supply Decoupling

As a rule, a suitable decoupling capacitor should be placed on each and every supply pin on all digital devices. It is important that these capacitors are placed as close to the chip supply pins as possible (less than 10 mm away). The ground connection of these capacitors should be tracked to 0 V by the shortest, heaviest traces possible.

Capacitors with a Type II dielectric, such as X5R or X7R and with a value of at least 100 nF, should be used for this purpose.

In addition, at least one ‘bulk’ decoupling capacitor, with a minimum value of 4.7 µF should be placed on each power rail, close to where the supply enters the board.

Surface mounting capacitors are preferred to wire leaded types due to their lower ESR and ESL. It is often possible to fit these decoupling capacitors underneath and on the opposite side of the PCB to the digital ICs. This will provide the shortest tracking, and most effective decoupling possible.

Refer to the application note *Selecting Decoupling Capacitors for Atmel PLDs* (doc0484.pdf; available on the Atmel website) for further general information on decoupling capacitors.

13.5 Single Supply Operation

When designing a PCB for an application using a single LDO, extra care should be taken to ensure short, low inductance traces between the supply and the touch controller supply input pins. Ideally, tracking for the individual supplies should be arranged in a star configuration, with the LDO at the junction of the star. This will ensure that supply current variations or noise in one supply rail will have minimum effect on the other supplies. In applications where a ground plane is not practical, this same star layout should also apply to the power supply ground returns.

13.6 Crystal Oscillator

If a crystal oscillator is used, its placement is critical to the performance of the design. The connecting leads between the device and the crystal should be as short as possible. These tracks, together with the crystal itself, should be placed above a suitable ground plane. It is also important that no other signal tracks are placed close to, or under, these tracks. The crystal input pins are at a relatively high impedance and cross-talk from other signals will seriously affect oscillator stability and accuracy. The crystal case should also be connected to ground if possible.

If an oscillator module is used, care still needs to be taken when tracking to the device. The clock signal should be kept as short as possible, with a solid ground return underneath the clock output.

13.7 Analog I/O

In general, tracking for the analog I/O signals from the device should be kept as short as possible. These normally go to a connector which interfaces directly to the touchscreen.

Ensure that adequate ground-planes are used. An analog ground plane should be used in addition to a digital one. Care should be taken to ensure that both ground planes are kept separate and are connected together only at the point of entry for the power to the PCB. This is usually at the input connector.

13.8 Component Placement and Tracking

It is important to orient all devices so that the tracking for important signals (such as power and clocks) are kept as short as possible. This simple point is often overlooked when initially planning a PCB layout and can save hours of work at a later stage.

13.8.1 Digital Signals

In general, when tracking digital signals, it is advisable to avoid sharp directional changes, sensitive signal tracks (such as analog I/O) and any clock or crystal tracking.

A good ground return path for all signals should be provided, where possible, to ensure that there are no discontinuities in the ground return path.

13.9 EMC and Other Observations

The following recommendations are not mandatory, but may help in situations where particularly difficult EMC or other problems are present:

- A small common mode choke is recommended on the differential USB data pair. This should be placed directly at the USB connector, between the connector and the relevant pins. Tracking lengths for the USB data pair should be kept as short as possible.
- Try to keep as many signals as possible on the inside layers of the board. If suitable ground flood fills are used on the top and bottom layers, these will provide a good level of screening for noisy signals, both into and out of the PCB.

- Ensure that the on-board regulators have sufficient tracking around and underneath the devices to act as a heatsink. This heatsink will normally be connected to the 0 V or ground supply pin. Increasing the width of the copper tracking to any of the device pins will aid in removing heat. There should be no solder mask over the copper track underneath the body of the regulators.
- Ensure that the decoupling capacitors, especially high capacity ceramic type, have the requisite low ESR, ESL and good stability/temperature properties. Refer to the regulator manufacturer’s datasheet for more information.

14. Getting Started with mXT2952T2

14.1 Establishing Contact

14.1.1 Communication with the Host

The host can use the following interface to communicate with the device:

- I^2C interface (see Section 10. on page 33)
- HID-I^2C interface (see Section 11. on page 39)
- USB interface (see Section 12. on page 49)

14.1.2 Power-up Sequence

On power-up, the $\overline{CHG}$ line goes low to indicate that there is new data to be read from the Message Processor T5 object. If the $\overline{CHG}$ line does not go low, there is a problem with the device.

The host should attempt to read any available messages to establish that the device is present and running following power-up or a reset. Examples of messages include reset or calibration messages. The host should also check that there are no configuration errors reported.

14.2 Using the Object Protocol

The device has an object-based protocol that is used to communicate with the device. Typical communication includes configuring the device, sending commands to the device, and receiving messages from the device. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information.

The host must perform the following initialization so that it can communicate with the device:

1. Read the start positions of all the objects in the device from the Object Table and build up a list of these addresses.
2. Use the Object Table to calculate the report IDs so that messages from the device can be correctly interpreted.

14.3 Writing to the Device

There are a number of mechanisms for writing to the device:

- Using an I^2C write operation (see Section 10.2 on page 33).
- Using the USB Generic HID write operation (see Section 12.3.1.2 on page 51).
- Using the Generic HID-I^2C write operation (see Section 11.5.1.2 on page 41).

To communicate with the device, you write to the appropriate object:

- To send a command to the device, you write the appropriate command to the Command Processor T6 object (refer to the *mXT2952T2 1.1 Protocol Guide*).
- To configure the device, you write to an object. For example, to configure the device power consumption you write to the global Power Configuration T7 object, and to set up a touchscreen you write to a Multiple Touch Touchscreen T100 object. Some objects are optional and need to be enabled before use. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on the objects.

14.4 Reading from the Device

Status information is stored in the Message Processor T5 object. This object can be read to receive any status information from the device. The following mechanisms provide an interrupt-style interface for reading messages in the Message Processor T5 object:

- The $\overline{CHG}$ line is asserted whenever a new message is available in the Message Processor T5 object (see Section 10.6 on page 36). See Section 10.4 on page 34 for information on the format of the I^2C read operation.
- When using the USB interface, the interface provides an interrupt-driven interface that sends the messages automatically (see Section 12.3.1.3 on page 52).
- When using the HID-I^2C interface, the interface provides an interrupt-driven interface that sends the messages automatically (see Section 11.5.1.3 on page 41)

Note that the host should always wait to be notified of messages. The host should not poll the device for messages.

14.5 Configuring the Device

The objects are designed such that a default value of zero in their fields is a “safe” value that typically disables functionality. The objects must be configured before use and the settings written to the nonvolatile memory using the Command Processor T6 object.

Perform the following actions for each object:

1. Enable the object, if the object requires it.
2. Configure the fields in the object, as required.
3. Enable reporting, if the object supports messages, to receive messages from the object.

Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on configuring the objects.

The following objects require no configuration:

- Debug Objects
  - Diagnostic Debug T37
- General objects:
  - Message Processor T5
  - Command Processor T6
- Support objects:
  - Message Count T44

The following objects must be configured before use:

- General objects
  - Power Configuration T7
  - Acquisition Configuration T8

The following objects should be checked and configured as necessary:

- Touch objects:
  - Key Array T15
  - Multiple Touch Touchscreen T100
- Signal processing objects:
  - Stylus T47
- Support objects:
  - Communications Configuration T18
  - GPIO/PWM Configuration T19
  - User Data T38
  - Digitizer HID Configuration T43
  - CTE Configuration T46
  - Self Capacitance Global Configuration T109
  - Self Capacitance Tuning Parameters T110
  - Self Capacitance Configuration T111
  - Self Capacitance Measurement Configuration T113

The following objects are optional and can be configured, as required:

- Signal processing objects:
  - Grip Suppression T40
  - Touch Suppression T42
  - Shieldless T56
  - Lens Bending T65
  - maXCharger T72

    - Glove Detection T78
    - Retransmission Compensation T80
    - Symbol Gesture Processor T92
    - Touch Sequence Processor T93
    - Self Capacitance maXCharger T108
- Support objects:
    - Self Test T25
    - Timer T61
    - Serial Data Command T68
    - Dynamic Configuration Controller T70
    - Dynamic Configuration Container T71
    - CTE Scan Configuration T77
    - Touch Event Trigger T79
    - Auxiliary Touch Configuration T104

15. Debugging

The device provides a mechanism for obtaining raw data for development and testing purposes by reading data from the Diagnostic Debug T37 object. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on this object.

A second mechanism is provided that allows the host to read the real-time raw data using the low-level debug port. This can be accessed via the SPI interface or the USB interface. Note that if both the I^2C-compatible and USB interfaces are used for normal communications, the debug data is output on the USB interface. Refer to QTAN0050, *Using the maXTouch Debug Port*, for more information on the debug port.

There is also a Self Test T25 object that runs self-test routines in the device to find hardware faults on the sense lines and the electrodes. This object also performs an initial pin fault test on power-up to ensure that there is no X-to-Y short before the high-voltage supply is enabled inside the chip. A high-voltage short into the analog circuitry would break the device.

Refer to the *mXT2952T2 1.1 Protocol Guide* and QTAN0059, *Using the maXTouch Self Test Feature*, for more information on the Self Test T25 object.

16. Specifications

16.1 Absolute Maximum Specifications

Vdd	3.6 V
VddIO	3.6 V
AVdd	3.6 V
XVdd	9.0 V
Voltage forced onto any pin	–0.3 V to (Vdd or AVdd) + 0.3 V
Configuration parameters maximum writes	10,000

CAUTION: Stresses beyond those listed under *Absolute Maximum Specifications* may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum specification conditions for extended periods may affect device reliability.

16.2 Recommended Operating Conditions

Operating temp	–40°C to +85°C
Storage temp	–60°C to +150°C
Vdd	3.3 V ±5%
VddIO	1.8 V to 3.3 V ±5% (I^2C Mode) 3.3 V ±5% (USB Mode)
AVdd	3.3 V ±5%
External XVdd – Static	8.5 V
XVdd – With Voltage Booster enabled (1)	6.15 V Nominal – Band Gap Referenced 7.38 V Nominal – Band Gap Referenced 8.51 V Nominal – Band Gap Referenced (Recommended)
Cx transverse load capacitance per node	0.5 pF to 3 pF
Temperature slew rate	10°C/min

Notes: 1. Refer to the *mXT2952T2 1.1 Protocol Guide* for more information on how to configure the XVdd voltage booster mode using the CTE Configuration T46 XVOLTAGE field.

16.2.1 Analog Voltage Supply

Parameter	Min	Typ	Max	Units	Notes
Operating limits	3.14	3.3	3.47	V	
Supply Rise Rates	–	–	0.36	V/µs	

16.2.2 Digital Voltage Supply

Parameter	Min	Typ	Max	Units	Notes
VddIO					
Operating limits	1.71	–	3.47	V	I^2C-compatible
	3.14	3.3	3.47	V	USB
Supply Rise Rates	–	–	0.36	V/µs	
Vdd					
Operating limits	3.14	3.3	3.47	V	
Supply Rise Rates	–	–	0.36	V/µs	

16.2.3 XVdd Supply

Parameter	Min	Typ	Max	Units	Notes
Operating limits	6.1	–	9.0	V	External XVdd supply
Supply Rise Rates	–	–	0.1	V/µs	

16.3 Test Configuration

The values listed below were used in the reference unit to validate the interfaces and derive the characterization data provided in the following sections.

See *mXT2952T2 1.1 Protocol Guide* for information about the individual objects and their fields.

The values for the user's application will depend on the circumstances of that particular project and will vary from those listed here. Further tuning will be required to achieve an optimal performance.

Table 16-1. Test Configuration

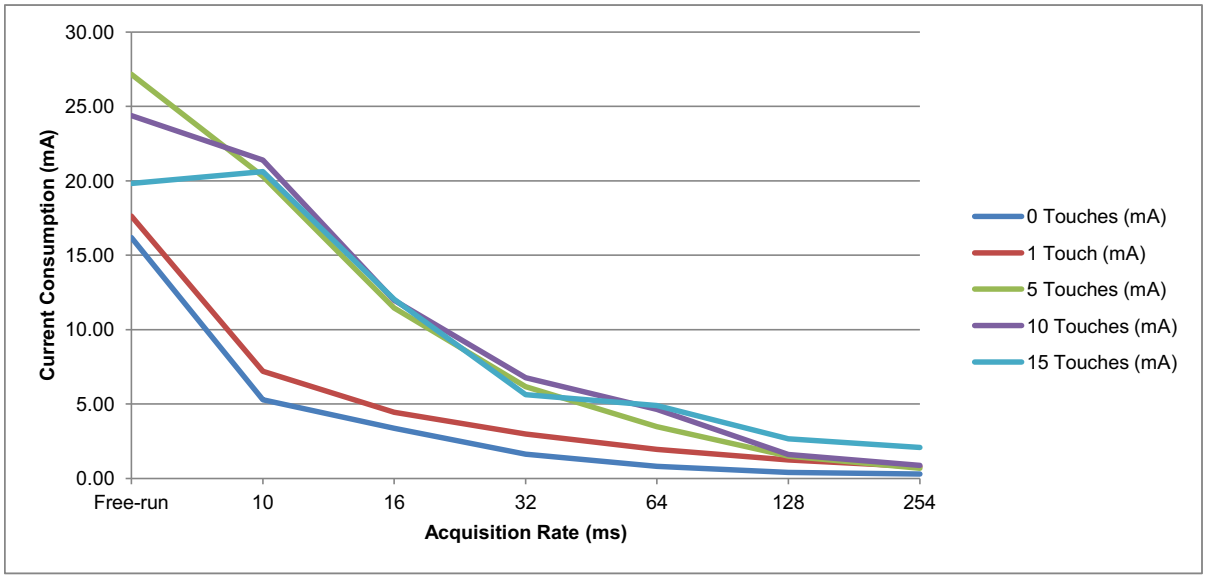
Object/Parameter	Description/Setting
GPIO/PWM Configuration T19	Object Enabled
Self Test T25	Object Enabled
Touch Suppression T42	Object Enabled
CTE Configuration T46	
IDLESYNCSPERX	8
ACTVSYNCSPERX	16
Stylus T47	Object Enabled
Shieldless T56	Object Enabled
Lens Bending T65	Object Enabled
Dynamic Configuration Controller T70	Object Enabled
maXCharger T72	Object Enabled
Touch Event Trigger T79	Object Enabled
Retransmission Compensation T80	Object Enabled
Multiple Touch Touchscreen T100	Object Enabled
XSIZE	41
YSIZE	71
Auxiliary Touch Configuration T104	Object Enabled
Self Capacitance maXCharger T108	Object Enabled
Self Capacitance Global Configuration T109	Object Enabled
Self Capacitance Measurement Configuration T113	Object Enabled

16.4 Supply Current – I²C Interface

16.4.1 Analog Supply

I2C Interface, AVdd = 3.287V

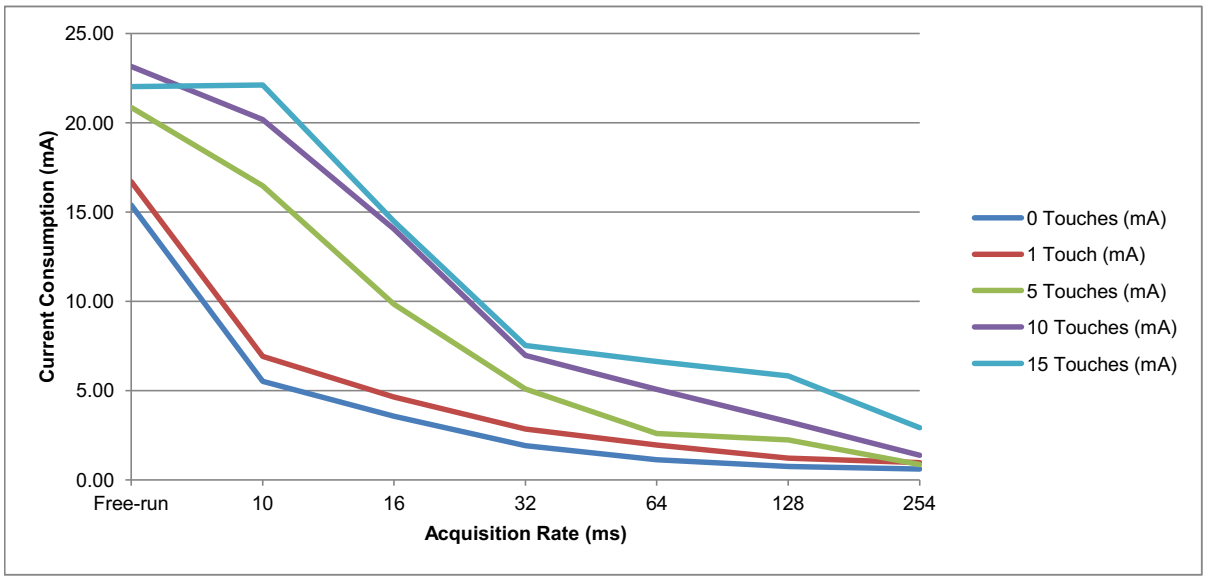
Acquisition Rate (ms)	0 Touches (mA)	1 Touch (mA)	5 Touches (mA)	10 Touches (mA)	15 Touches (mA)
Free-run	16.19	17.64	27.16	24.38	19.83
10	5.29	7.20	20.29	21.39	20.63
16	3.36	4.45	11.46	11.98	12.03
32	1.64	2.98	6.17	6.76	5.62
64	0.81	1.96	3.47	4.64	4.90
128	0.39	1.25	1.50	1.61	2.65
254	0.28	0.79	0.69	0.87	2.08



16.4.2 Digital Supply

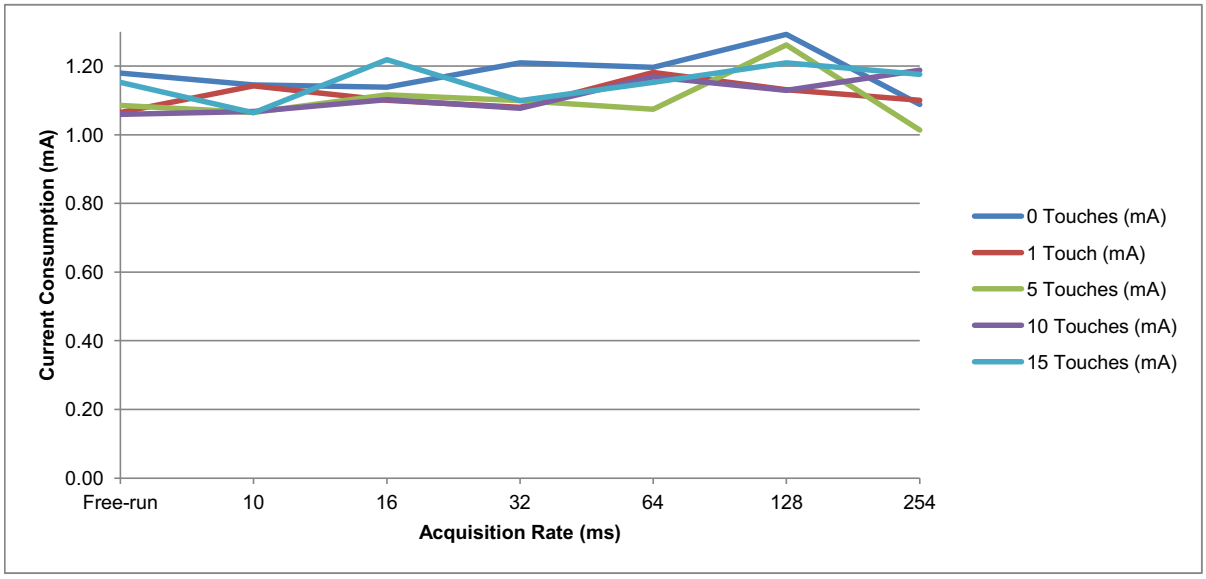
I2C Interface, Vdd = 3.298

Acquisition Rate (ms)	0 Touches (mA)	1 Touch (mA)	5 Touches (mA)	10 Touches (mA)	15 Touches (mA)
Free-run	15.39	16.69	20.85	23.15	22.02
10	5.51	6.91	16.46	20.18	22.11
16	3.56	4.64	9.83	14.04	14.47
32	1.92	2.85	5.08	6.96	7.52
64	1.12	1.94	2.59	5.07	6.63
128	0.74	1.21	2.24	3.26	5.81
254	0.61	0.97	0.85	1.37	2.91



I2C Interface, VddIO = 3.299V

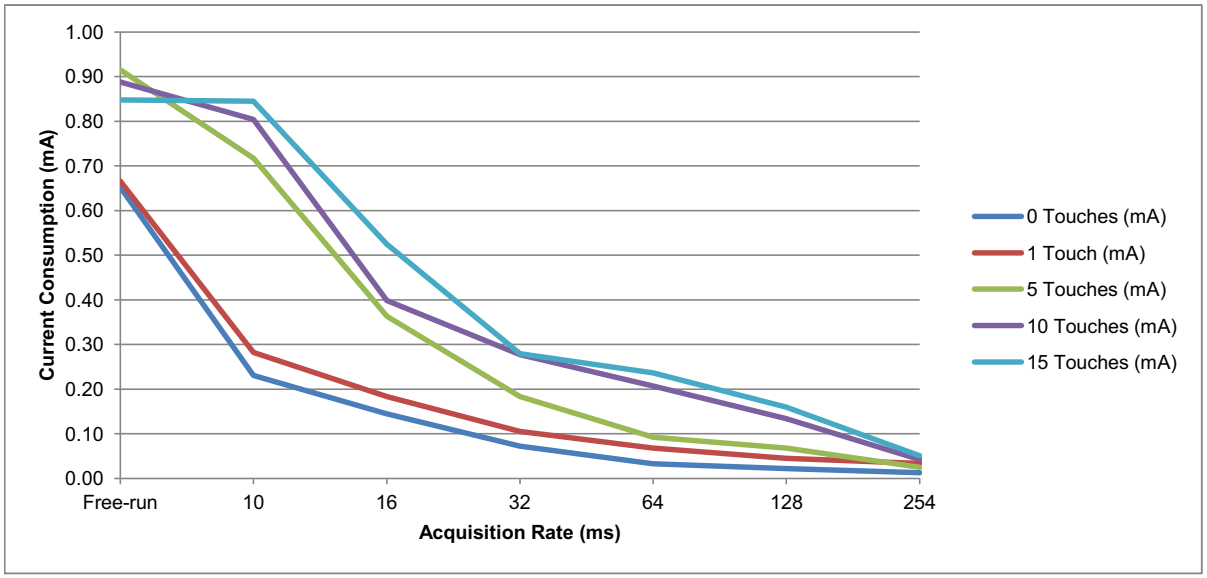
Acquisition Rate (ms)	0 Touches (mA)	1 Touch (mA)	5 Touches (mA)	10 Touches (mA)	15 Touches (mA)
Free-run	1.18	1.07	1.09	1.06	1.15
10	1.15	1.14	1.07	1.07	1.06
16	1.14	1.10	1.12	1.10	1.22
32	1.21	1.08	1.10	1.08	1.10
64	1.20	1.18	1.07	1.17	1.15
128	1.29	1.13	1.26	1.13	1.21
254	1.09	1.10	1.01	1.19	1.18



16.4.3 XVdd Supply

I2C Interface, XVdd = 8.55V

Acquisition Rate (ms)	0 Touches (mA)	1 Touch (mA)	5 Touches (mA)	10 Touches (mA)	15 Touches (mA)
Free-run	0.65	0.67	0.92	0.89	0.85
10	0.23	0.28	0.72	0.80	0.84
16	0.14	0.18	0.36	0.40	0.52
32	0.07	0.11	0.18	0.28	0.28
64	0.03	0.07	0.09	0.21	0.24
128	0.02	0.05	0.07	0.13	0.16
254	0.01	0.03	0.02	0.04	0.05

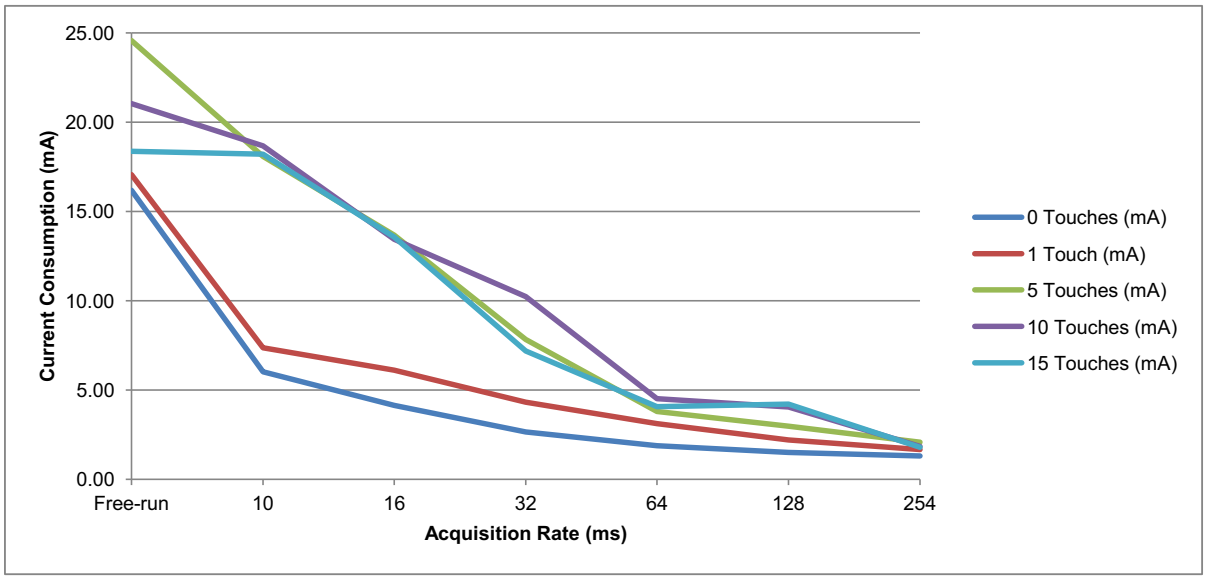


16.5 Supply Current – USB Interface

16.5.1 Analog Supply

USB Interface, AVdd = 3.287V

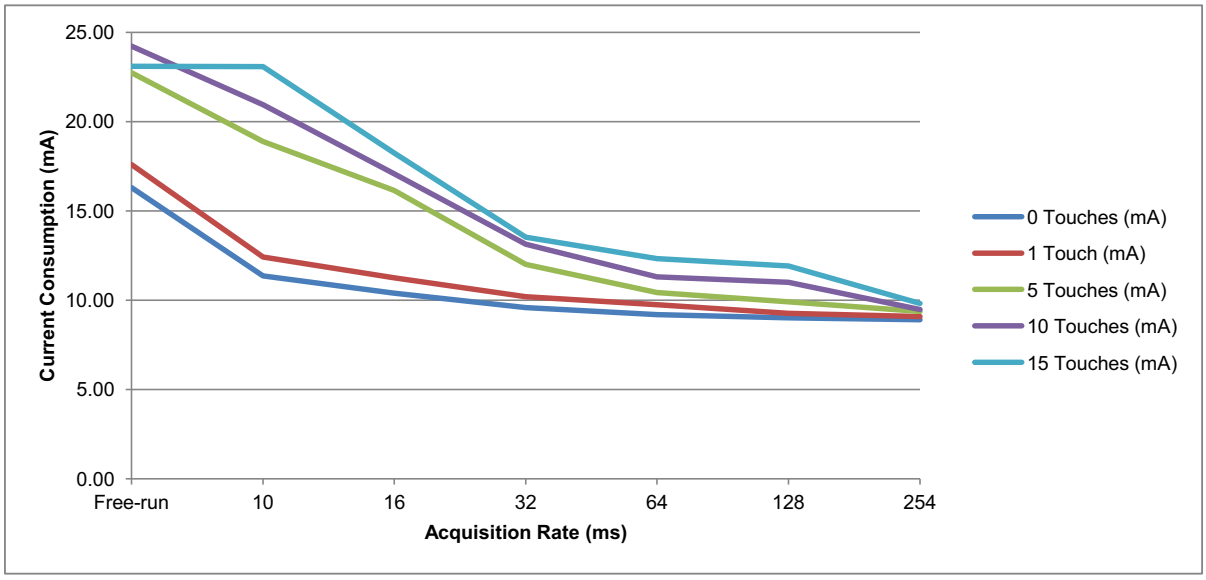
Acquisition Rate (ms)	0 Touches (mA)	1 Touch (mA)	5 Touches (mA)	10 Touches (mA)	15 Touches (mA)
Free-run	16.17	17.05	24.58	21.04	18.37
10	6.01	7.36	18.08	18.67	18.20
16	4.14	6.10	13.67	13.43	13.55
32	2.64	4.31	7.83	10.22	7.18
64	1.88	3.11	3.79	4.51	4.06
128	1.50	2.21	2.96	4.05	4.21
254	1.31	1.66	2.07	1.84	1.79



16.5.1.1 Digital Supply

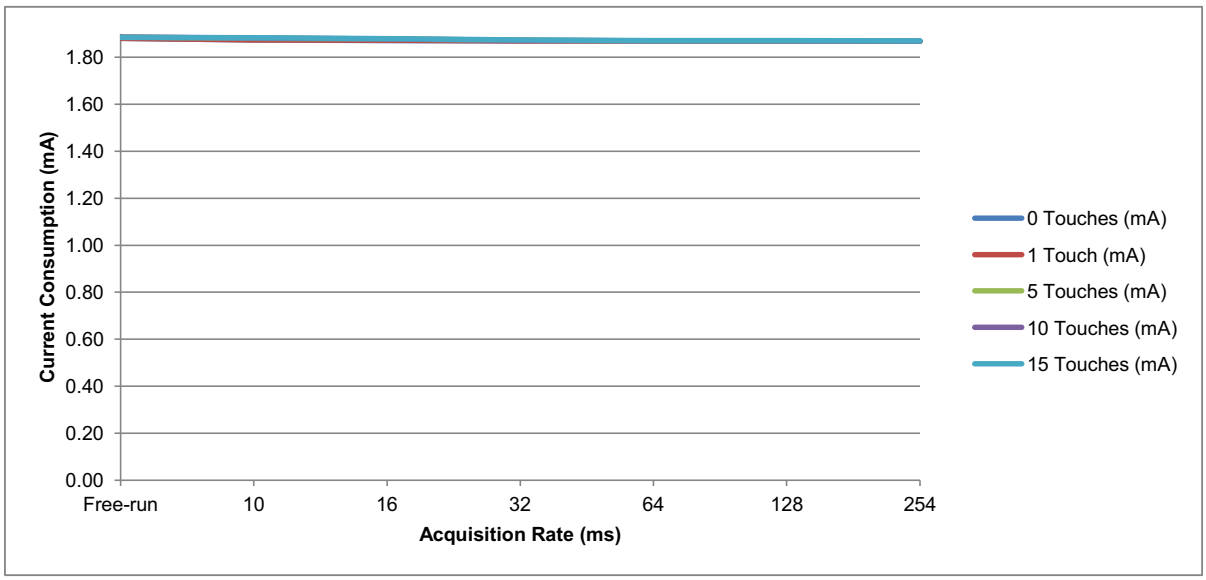
USB Interface, Vdd = 3.298

Acquisition Rate (ms)	0 Touches (mA)	1 Touch (mA)	5 Touches (mA)	10 Touches (mA)	15 Touches (mA)
Free-run	16.30	17.59	22.75	24.23	23.09
10	11.37	12.41	18.88	20.95	23.09
16	10.39	11.25	16.15	17.07	18.23
32	9.58	10.19	12.01	13.14	13.53
64	9.20	9.74	10.43	11.30	12.33
128	9.01	9.26	9.90	11.00	11.92
254	8.90	9.08	9.37	9.47	9.82



USB Interface, VddIO = 3.299V

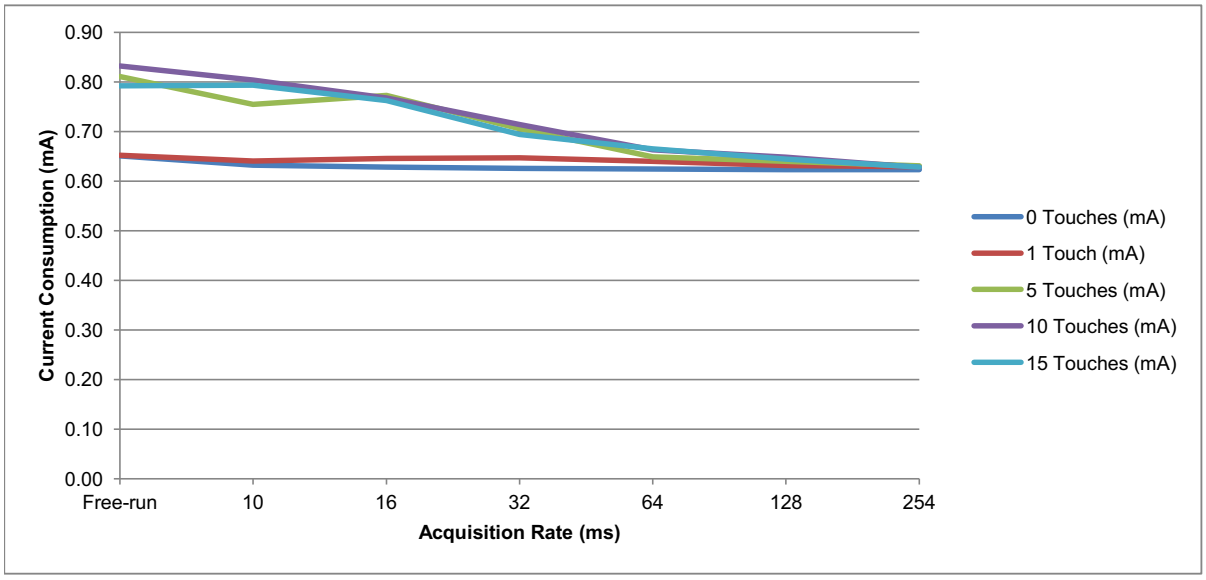
Acquisition Rate (ms)	0 Touches (mA)	1 Touch (mA)	5 Touches (mA)	10 Touches (mA)	15 Touches (mA)
Free-run	1.88	1.88	1.89	1.89	1.88
10	1.87	1.87	1.88	1.88	1.88
16	1.87	1.87	1.88	1.88	1.88
32	1.87	1.87	1.87	1.87	1.87
64	1.87	1.87	1.87	1.87	1.87
128	1.87	1.87	1.87	1.87	1.87
254	1.87	1.87	1.87	1.87	1.87



16.5.1.2 XVdd Supply

USB Interface, XVdd = 8.55V

Acquisition Rate (ms)	0 Touches (mA)	1 Touch (mA)	5 Touches (mA)	10 Touches (mA)	15 Touches (mA)
Free-run	0.65	0.65	0.81	0.83	0.79
10	0.63	0.64	0.75	0.80	0.79
16	0.63	0.65	0.77	0.77	0.76
32	0.63	0.65	0.71	0.71	0.69
64	0.62	0.64	0.65	0.66	0.66
128	0.62	0.63	0.64	0.65	0.64
254	0.62	0.63	0.63	0.63	0.63



16.6 Deep Sleep Current

T_A = 25°C

Parameter	Min	Typ	Max	Units	Notes
Deep Sleep Current	–	720	–	µA	Vdd = 3.3 V, AVdd = 3.3 V
Deep Sleep Power	–	2376	–	µW	Vdd = 3.3 V, AVdd = 3.3 V

16.7 Power Supply Ripple and Noise

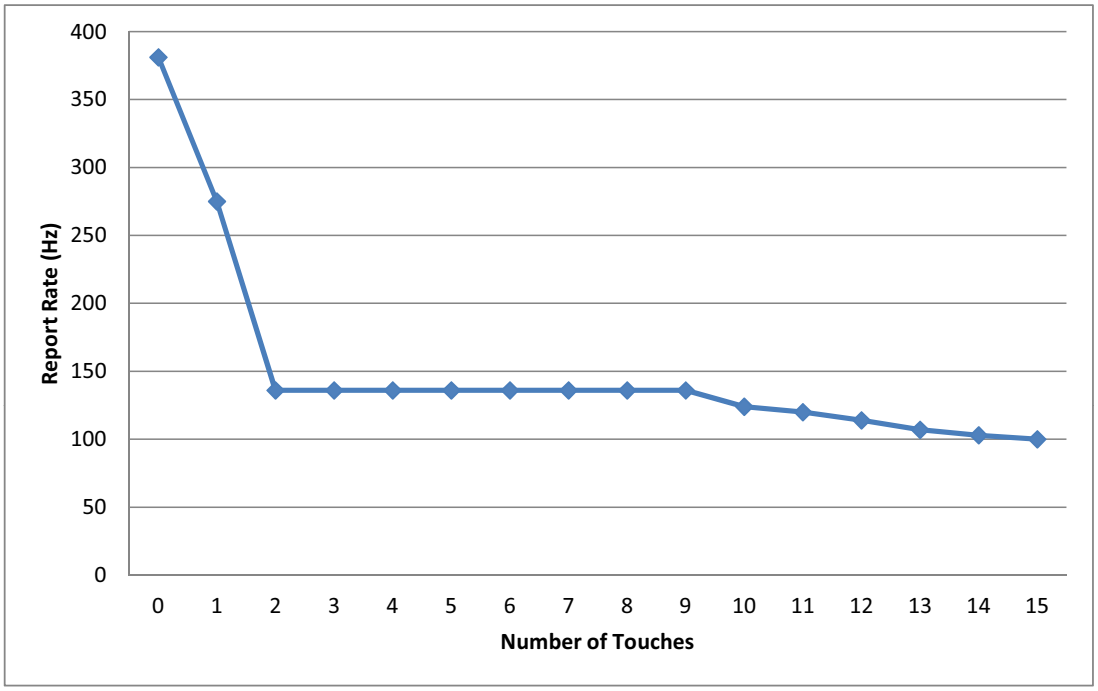
Parameter	Min	Typ	Max	Units	Notes
Vdd	–	–	±50	mV	Across frequency range 1 Hz to 1 MHz
AVdd	–	–	±25	mV	Across frequency range 1 Hz to 1 MHz
AVdd (with noise suppression enabled)	–	–	±40	mV	Across frequency range 1 Hz to 1 MHz, with Noise Suppression enabled

16.8 Timing Specifications

16.8.1 Touch Latency

Description	Min	Typ	Max	Units	Notes
100 Hz	–	23.7	–	ms	

16.8.2 Speed



16.8.3 Reset Timing

Parameter	Min	Typ	Max	Units	Notes
Power on to $\overline{\text{CHG}}$ line low	–	100	–	ms	Vdd supply for POR VddIO supply for external reset
Hardware reset to $\overline{\text{CHG}}$ line low	–	100	–	ms	
Software reset to $\overline{\text{CHG}}$ line low	–	101	–	ms	

The mXT2952T2 meets Microsoft Windows 8 requirements.

Note: Any $\overline{\text{CHG}}$ line activity before the power-on or reset period has expired should be ignored by the host. Operation of this signal cannot be guaranteed before the power-on/reset periods have expired (see Table 16.8.3).

16.9 Input/Output Characteristics

Parameter	Description	Min	Typ	Max	Units	Notes
Input ($\overline{\text{RESET}}$, GPIO, SDA, SCL)						
Vil	Low input logic level	–0.3	–	0.3 × VddIO	V	VddIO = 1.8 V to Vdd
Vih	High input logic level	0.7 × VddIO	–	VddIO	V	VddIO = 1.8 V to Vdd
Iil	Input leakage current	–	–	1	µA	Pull-up resistors disabled
$\overline{\text{RESET}}$ pin	Internal pull-up resistor	20	40	60	kΩ	
Output ($\overline{\text{CHG}}$, GPIO)						
Vol	Low output voltage	0	–	0.2 × VddIO	V	VDDIO = 1.8 V to VDD. Iol = -2mA
Voh	High output voltage	0.8 × VddIO	–	VddIO	V	VDDIO = 1.8 V to VDD. Ioh = 2mA

16.10 I²C Specifications

Parameter	Value
Addresses	0x4A or 0x4B
Maximum bus speed (SCL)	3.4 MHz
I²C specification	Version 6.0
Required pull-up resistance for standard mode (100 kHz)	1 kΩ to 10 kΩ (1)
Required pull-up resistance for fast mode (400 kHz)	1 kΩ to 3 kΩ (1)
Required pull-up resistance for fast+ mode (1 MHz)	0.7 kΩ (max) (1)
Required pull-up resistance for high-speed mode (3.4 MHz)	0.5 kΩ to 0.75 kΩ (1)

Notes: 1. The values of pull-up resistors should be chosen to ensure SCL and SDA rise and fall times meet the I²C specification. The value required will depend on the amount of stray capacitance on the line.

16.11 USB Specification

Parameter	Value
Endpoint Addresses	0x81 (Endpoint 1) 0x02 (Endpoint 2) 0x83 (Endpoint 3)
Maximum bus speed	12 Mbps
Vendor ID	0x03EB (Atmel)
Product ID	0x214E (mXT2952T2)
USB specification	USB 2.0 HID specification 1.11 with amendments for multitouch digitizers

16.12 HID-I^{2}C Specification

Parameter	Value
Vendor ID	0x03EB (Atmel)
Product ID	0x214E (mXT2952T2)
HID-I^{2}C specification	1.0

16.13 Touch Accuracy and Repeatability

Parameter	Min	Typ	Max	Units	Notes
Linearity (touch only; 5.4 mm electrode pitch)	–	±1	–	mm	8 mm or greater finger
Linearity (touch only; 4.2 mm electrode pitch)	–	±0.5	–	mm	4 mm or greater finger
Accuracy	–	±1	–	mm	
Accuracy at edge	–	±2	–	mm	
Repeatability	–	±0.25	–	%	X axis with 12-bit resolution

16.14 Thermal Packaging

16.14.1 Thermal Data

Parameter	Typ	Unit	Condition	Package
Junction to ambient thermal resistance	46.2	°C/W	Still air	UFBGA 162, 10 × 5 mm
Junction to case thermal resistance	8.7	°C/W		UFBGA 162, 10 × 5 mm

16.14.2 Junction Temperature

The average chip junction temperature, T_J in °C can be obtained from the following:

$$T_J = T_A + (P_D \times \theta_{JA})$$

If a cooling device is required, use this equation:

$$T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$$

where:

- θ_{JA}= package thermal resistance, Junction to ambient (°C/W).
- θ_{JC} = package thermal resistance, Junction to case thermal resistance (°C/W).
- $\theta_{HEATSINK}$ = cooling device thermal resistance (°C/W), provided in the cooling device datasheet.
- P_D = device power consumption (W).
- T_A is the ambient temperature (°C).

16.15 ESD Information

Parameter	Value	Reference standard
Human Body Model (HBM)	±2000 V	JEDEC JS–001
Charge Device Model (CDM)	±250 V	

16.16 Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/s max
Preheat Temperature 175°C ±25°C	150 – 200°C
Time Maintained Above 217°C	60 – 150 s
Time within 5°C of Actual Peak Temperature	30 s
Peak Temperature Range	260°C
Ramp down Rate	6°C/s max
Time 25°C to Peak Temperature	8 minutes max

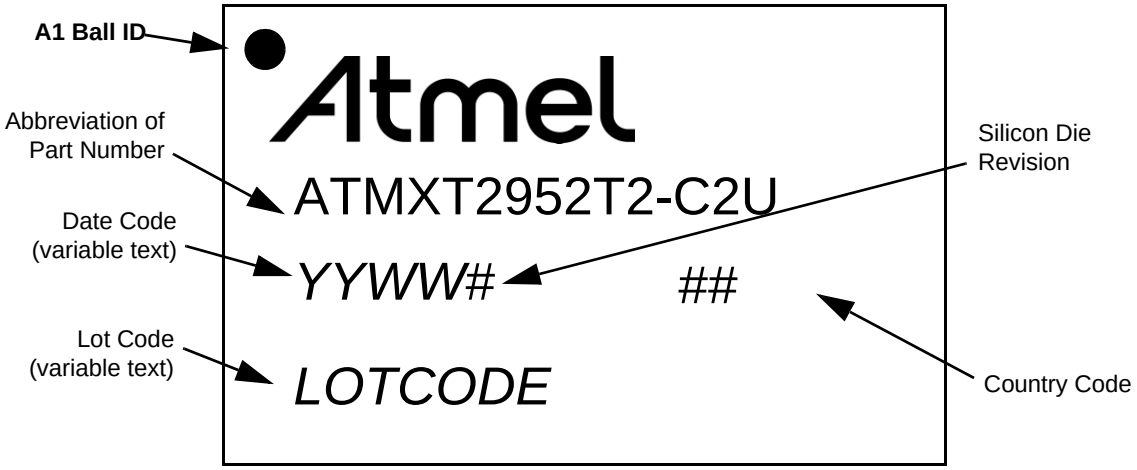
16.17 Moisture Sensitivity Level (MSL)

MSL Rating	Package Type(s)	Peak Body Temperature	Specifications
MSL3	BGA	260°C	IPC/JEDEC J-STD-020

17. Package Information

17.1 Part Marking

17.1.1 ATMXT2952T2-C2U – 162-ball UFBGA



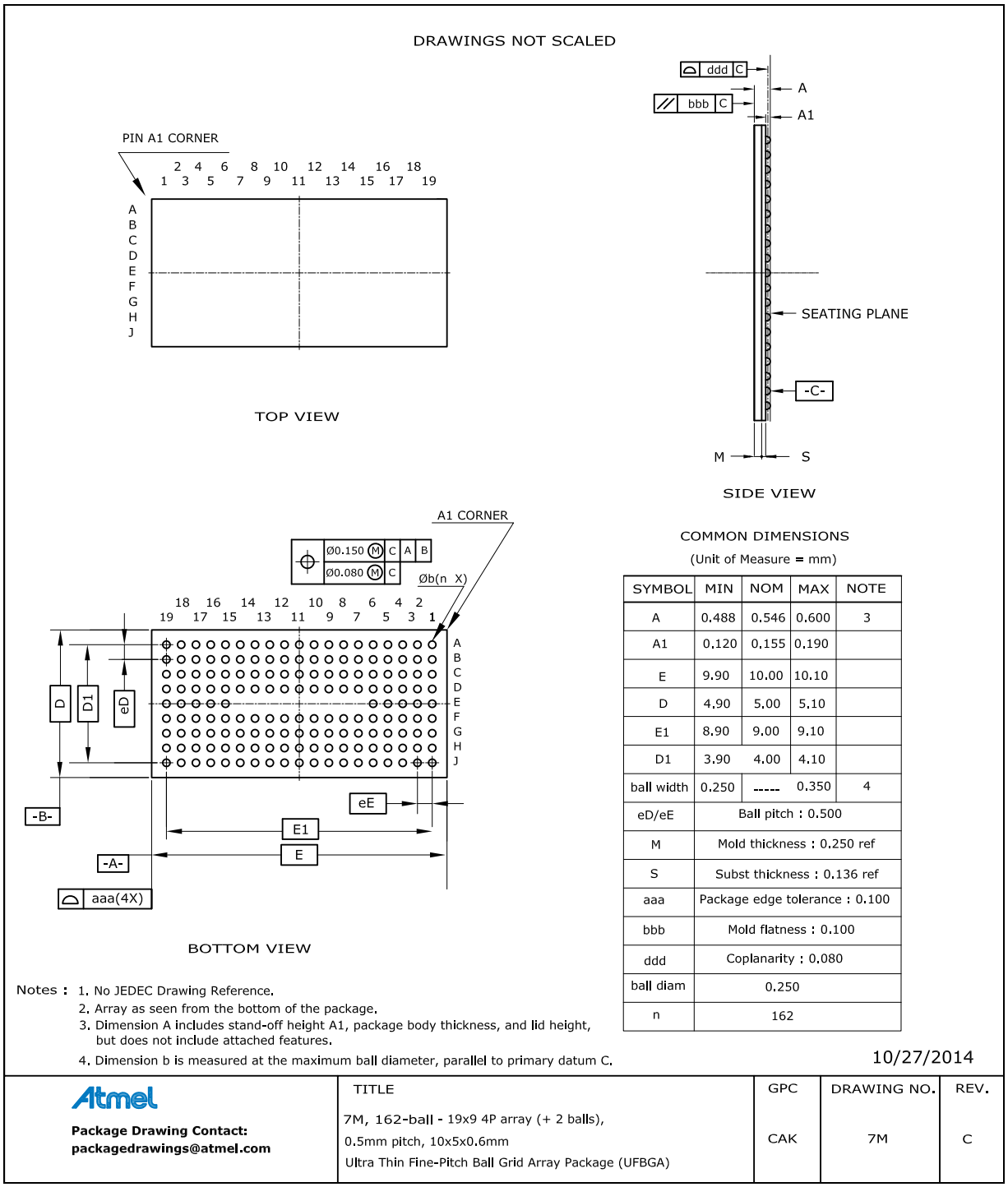
17.2 Orderable Part Number

Orderable Part Number	QS Number	Firmware Revision	Silicon Die Revision[1]	Description
ATMXT2952T2-C2U (Supplied in trays)	959	1.1	D	162-ball 10 × 5 × 0.6 mm, 0.5 mm pitch UFBGA, RoHS compliant
ATMXT2952T2-C2UR (Supplied in tapes and reels)	959	1.1	D	162-ball 10 × 5 × 0.6 mm, 0.5 mm pitch UFBGA, RoHS compliant

Notes: 1. Refer to Application Note *MXTAN0205: Reading the Revision ID on T2 Devices* for information on the Silicon Die Revision

17.3 Mechanical Drawings

17.3.1 UFBGA 162 Balls



COMMON DIMENSIONS

(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.488	0.546	0.600	3
A1	0.120	0.155	0.190	
E	9.90	10.00	10.10	
D	4.90	5.00	5.10	
E1	8.90	9.00	9.10	
D1	3.90	4.00	4.10	
ball width	0.250	-----	0.350	4
eD/eE	Ball pitch : 0.500			
M	Mold thickness : 0.250 ref			
S	Subst thickness : 0.136 ref			
aaa	Package edge tolerance : 0.100			
bbb	Mold flatness : 0.100			
ddd	Coplanarity : 0.080			
ball diam	0.250			
n	162			

Notes :
1. No JEDEC Drawing Reference.
2. Array as seen from the bottom of the package.
3. Dimension A includes stand-off height A1, package body thickness, and lid height, but does not include attached features.
4. Dimension b is measured at the maximum ball diameter, parallel to primary datum C.

10/27/2014

	TITLE	GPC	DRAWING NO.	REV.
Atmel **Package Drawing Contact:** **packagedrawings@atmel.com**	7M, 162-ball - 19x9 4P array (+ 2 balls), 0.5mm pitch, 10x5x0.6mm Ultra Thin Fine-Pitch Ball Grid Array Package (UFBGA)	CAK	7M	C

Appendix A. QMatrix Primer

A.1 Acquisition Technique

QMatrix capacitive acquisition uses a series of pulses to deposit charge into a sampling capacitor, Cs. The pulses are driven on X lines from the controller. The rising edge of the pulse causes current to flow in the mutual capacitance, Cx, formed between the X line and a neighboring receiver electrode or Y line. While one X line is being pulsed, all others are grounded. This leads to excellent isolation of the particular mutual capacitances being measured [1], a feature that makes for good inherent touchscreen performance.

After a fixed number of pulses (known as the burst length) the sampling capacitor's voltage is measured to determine how much charge has accumulated. This charge is directly proportional to Cx and therefore changes if Cx [2] changes. The transmit-receive charge transfer process between the X lines and Y lines causes an electric field to form that loops from X to Y. The field itself emanates from X and terminates on Y. If the X and Y electrodes are fixed directly [3] to a dielectric material like plastic or glass, then this field tends to channel through the dielectric with very little leakage of the field out into free-space (that is, above the panel). Some proportion of the field does escape the surface of the dielectric, however, and so can be influenced during a touch.

When a finger is placed in close proximity (a few millimeters) or directly onto the dielectric's surface, some of this stray field and some of the field that would otherwise have propagated via the dielectric and terminated onto the Y electrode, is diverted into the finger and is conducted back to the controller chip via the human body rather than via the Y line.

This means that less charge is accumulated in Cs, and hence the terminal voltage present on Cs, after all the charge transfer pulses are complete, becomes less. In this way, the controller can measure changes in Cx during touch. This means that the measured capacitance Cx goes down during touch, because the coupled field is partly diverted by the touching object.

The spatial separation between the X and Y electrodes is significant to make the electric field to propagate well in relation to the thickness of the dielectric panel.

A.2 Moisture Resistance

A useful side effect of the QMatrix acquisition method is that placing a floating conductive element between the X and Y lines tends to increase the field coupling and so increases the capacitance Cx. This is the opposite change direction to normal touch, and so can be quite easily ignored or compensated for by the controller. An example of such floating conductive elements is the water droplets caused by condensation.

As a result, QMatrix-based touchscreens tend not to go into false detect when they are covered in small non-coalesced water droplets. Once the droplets start to merge, however, they can become large enough to bridge the field across to nearby ground return paths (for example, other X lines not currently driven, or ground paths in mechanical chassis components). When this happens, the screen's behavior can become erratic.

There are some measures used in these controllers to help with this situation, but in general there comes a point where the screen is so contaminated by moisture that false detections become inevitable. It should also be noted that uniform condensation soon becomes non-uniform once a finger has spread it around. Finger grease renders the water highly conductive, making the situation worse overall.

In general, QMatrix has industry-leading moisture tolerance but there comes a point when even the best capacitive touchscreen suffers due to moisture on the dielectric surface.

1. A common problem with other types of capacitive acquisition technique when used for touchscreens, is that this isolation is not so pronounced. This means that when touching one region of the screen, the capacitive signals also tend to change slightly in nearby nodes too, causing small but often significant errors in the reported touch position.
2. To a first approximation.
3. Air gaps in front of QMatrix sensors massively reduce this field propagation and kill sensitivity. Normal optically clear adhesives work well to attach QMatrix touchscreens to their dielectric front panel.

A.3 Interference Sources

A.3.1 Power Supply

The device can tolerate short-term power supply fluctuations. If the power supply fluctuates slowly with temperature, the device tracks and compensate for these changes automatically with only minor changes in sensitivity. If the supply voltage drifts or shifts quickly, the drift compensation mechanism is not able to keep up, causing sensitivity anomalies or false detections.

If power supply noise is present (usually caused by LEDs, relays, or other high current devices) and affects the measurement results then a separate Low Dropout (LDO) type regulator should be used for the AVdd power supply.

It is recommended that all ceramic decoupling capacitors on supply lines are placed very close (<5 mm) to the chip. A bulk capacitor of at least 2.2 µF and a higher frequency capacitor of around 10 nF to 100 nF in parallel are recommended; both must be X7R or X5R dielectric capacitors.

A.3.2 Other Noise Sources

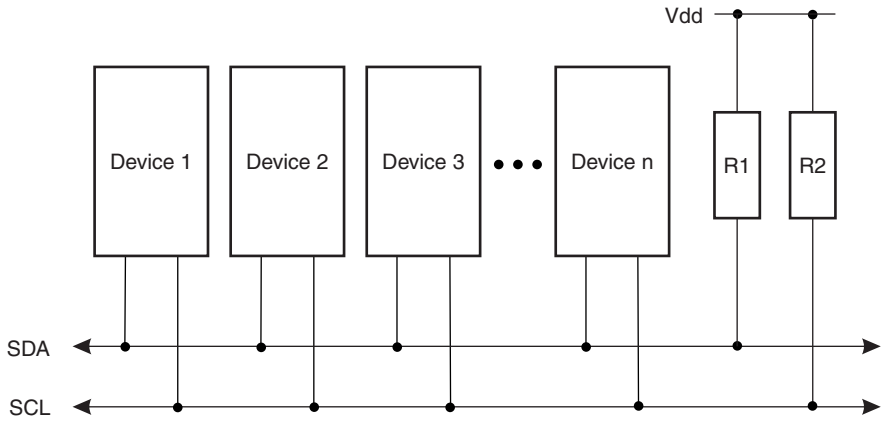
Refer to QTAN0079, *Buttons, Sliders and Wheels Sensor Design Guide*, for information (downloadable from the Touch Technology area of the Atmel website).

Appendix B. I²C Basics (I²C Operation)

B.1 Interface Bus

The device communicates with the host over an I²C bus. The following sections give an overview of the bus; more detailed information is available from www.nxp.com/documents/user_manual/UM10204.pdf. Devices are connected to the I²C bus as shown in Figure B-1. Both bus lines are connected to Vdd via pull-up resistors. The bus drivers of all I²C devices must be open-drain type. This implements a wired AND function that allows any and all devices to drive the bus, one at a time. A low level on the bus is generated when a device outputs a zero.

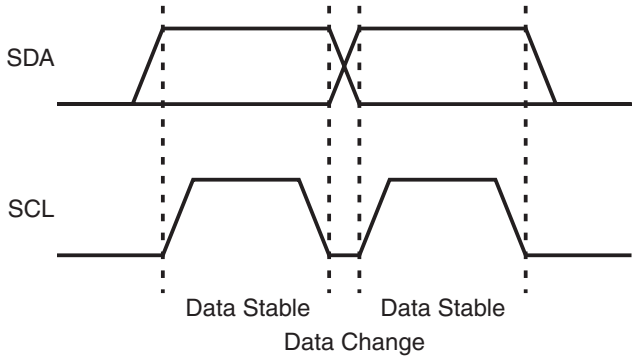
Figure B-1. I²C Interface Bus



B.2 Transferring Data Bits

Each data bit transferred on the bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high; the only exception to this rule is for generating START and STOP conditions.

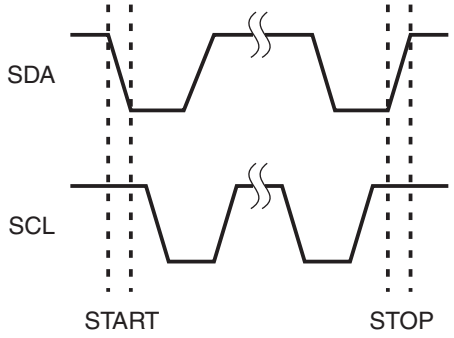
Figure B-2. Data Transfer



B.3 START and STOP Conditions

The host initiates and terminates a data transmission. The transmission is initiated when the host issues a START condition on the bus, and is terminated when the host issues a STOP condition. Between the START and STOP conditions, the bus is considered busy. As shown in Figure B-3, START and STOP conditions are signaled by changing the level of the SDA line when the SCL line is high.

Figure B-3. START and STOP Conditions

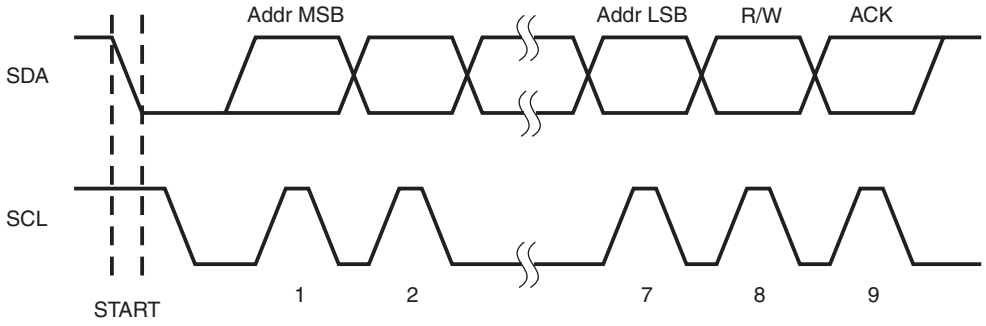


B.4 Address Byte Format

All address bytes are 9 bits long, consisting of 7 address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is performed, otherwise a write operation is performed. When the device recognizes that it is being addressed, it will acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. An address byte consisting of a slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The most significant bit of the address byte is transmitted first. The address sent by the host must be consistent with that selected with the option jumpers.

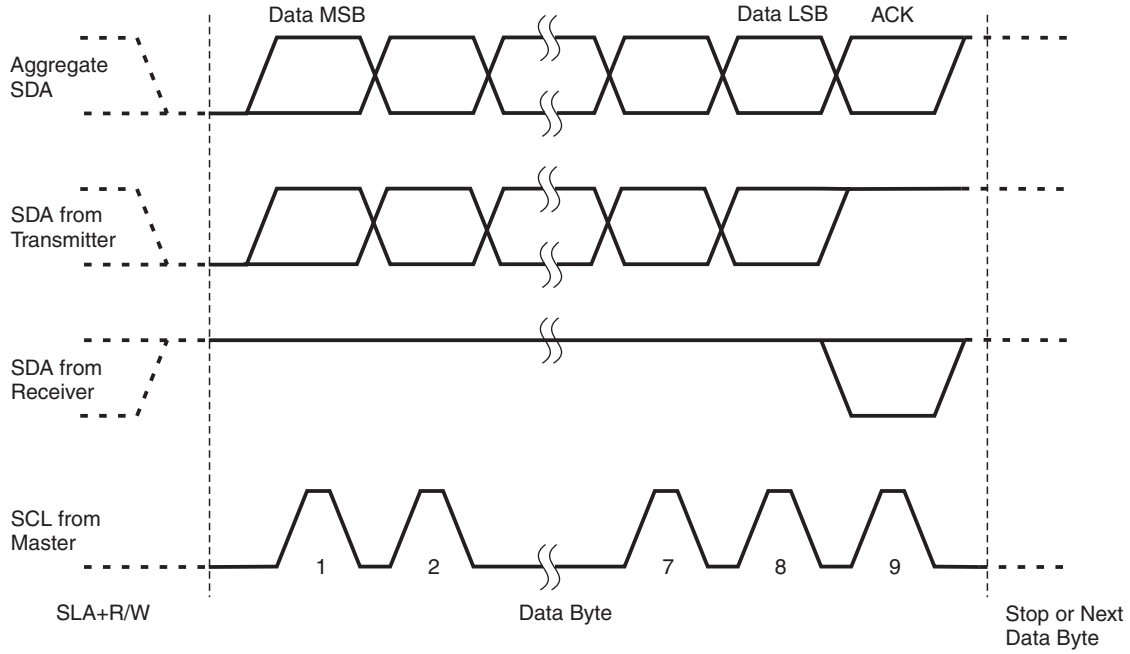
Figure B-4. Address Byte Format



B.5 Data Byte Format

All data bytes are 9 bits long, consisting of 8 data bits and an acknowledge bit. During a data transfer, the host generates the clock and the START and STOP conditions, while the receiver is responsible for acknowledging the reception. An acknowledge (ACK) is signaled by the receiver pulling the SDA line low during the ninth SCL cycle. If the receiver leaves the SDA line high, a NACK is signaled.

Figure B-5. Data Byte Format



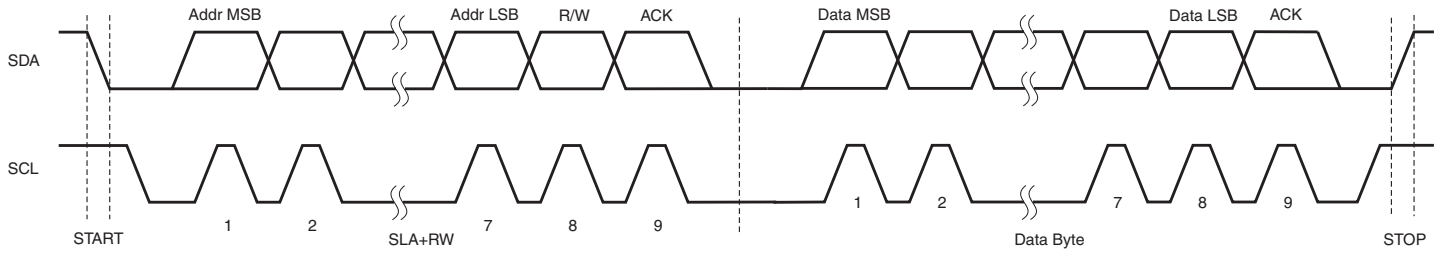
B.6 Combining Address and Data Bytes into a Transmission

A transmission consists of a START condition, an SLA+R/W, one or more data bytes and a STOP condition. The wired “ANDing” of the SCL line is used to implement handshaking between the host and the device. The device extends the SCL low period by pulling the SCL line low whenever it needs extra time for processing between the data transmissions.

Note: Each write or read cycle must end with a stop condition. The device may not respond correctly if a cycle is terminated by a new start condition.

Figure B-6 shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP.

Figure B-6. Byte Transmission



Appendix C. Glossary of Terms

Channel

See **Node**.

Jitter

The peak-to-peak variance in the reported location for an axis when a fixed touch is applied. Typically jitter is random in nature and has a Gaussian [1] distribution, therefore measurement of peak-to-peak jitter must be conducted over some period of time, typically a few seconds. Jitter is typically measured as a percentage of the axis in question.

For example a 100 × 100 mm Touchscreen that shows ±0.5% jitter in X and ±1% jitter in Y would show a peak deviation from the average reported coordinate of ±0.5 mm in X and ±1 mm in Y. Note that by defining the jitter relative to the average reported coordinate, the effects of linearity are ignored.

Linearity

The measurement of the peak-to-peak deviation of the reported touch coordinate in one axis relative to the absolute position of touch on that axis. This is often referred to as the nonlinearity. Non-linearity in either X or Y axes manifest themselves as regions where the perceived touch motion along that axis (alone) is not reflected correctly in the reported coordinate giving the sense of moving too fast or too slow. Linearity is measured as a percentage of the axis in question.

For each axis, a plot of the true coordinate versus the reported coordinate should be a perfect straight line at 45°. A non-linearity makes this plot deviate from this ideal line. It is possible to correct modest non-linearity using on-chip linearization tables, but this correction trades linearity for resolution in regions where stronger corrections are needed (because there is a stretching or compressing effect to correct the nonlinearity, so altering the resolution in these regions). Linearity is typically measured using data that has been sufficiently filtered to remove the effects of jitter. For example, a 100 mm slider with a nonlinearity of ±1% reports a position that is, at most, 1 mm away in either direction from the true position.

Multitouch

The ability of a touchscreen to report multiple concurrent touches. The touches are reported as separate sets of XY co-ordinates.

Node

One of the capacitive measurement points at which the sensor controller can detect capacitive change.

Resolution

The measure of the smallest movement on a slider or touchscreen in an axis that causes a change in the reported coordinate for that axis. Resolution is normally expressed in bits and tends to refer to resolution across the whole axis in question. For example, a resolution of 10 bits can resolve a movement of 0.0977 mm on a slider 100 mm long. Jitter in the reported position degrades usable resolution.

Touchscreen

A two-dimensional arrangement of electrodes whose capacitance changes when touched, allowing the location of touch to be computed in both X and Y axes. The output from the XY computation is a pair of numbers, typically 12-bits each, ranging from 0 to 4095, representing the extents of the touchscreen active region.

1. Sometimes called Bell-shaped or Normal distribution.

Associated Documents

Note: The documents listed below are available under NDA only. In addition, some documents may have further restrictions placed upon them.

For information on using and configuring the device, see the following:

- *mXT2952T2 1.1 Protocol Guide* (distributed with Atmel approval only)

The following documents may also be useful (available by contacting Atmel Touch Technology Division):

- **Touchscreen design and PCB/FPCB layout guidelines:**
  - Application Note: QTAN0054 – *Getting Started with maXTouch Touchscreen Designs*
  - Application Note: MXTAN0208 – *Design Guide for PCB Layouts for Atmel Touch Controllers*
  - Application Note: QTAN0080 – *Touchscreens Sensor Design Guide*
- **Configuring the device:**
  - Application Note: QTAN0078 – *maXTouch Stylus Tuning*
  - Application Note: QTAN0059 – *Using the maXTouch Self Test Feature*
  - Application Note: QTAN0070 – *Recovering from Palm Touches During Calibration with maXTouch Touchscreen Controllers*
- **Miscellaneous:**
  - Application Note: QTAN0050 – *Using the maXTouch Debug Port*
  - Application Note: QTAN0058 – *Rejecting Unintentional Touches with the maXTouch Touchscreen Controllers*
  - Application Note: QTAN0061 – *maXTouch Sensitivity Effects for Mobile Devices*
  - Application Note: QTAN0083 – *Power and Speed Considerations*
  - Application Note: QTAN0051 – *Bootloading Procedure for Atmel Touch Sensors Based on the Object Protocol*
- **Tools:**
  - QTAN0101 – Object Server User Guide

Revision History

Revision Number	History
Revision AX – July 2014	Initial edition for firmware revision 1.1.AB – Preliminary
Revision BX – September 2014	Updated for firmware revision 1.1.AC – Release
Revision CX – December 2014	• Corrected reset value • Revised component labeling • Revised voltage regulator recommendations • Revised description of USB interface (no functional change)
Revision DX – February 2015	• Updated Power-up / Reset Requirements section

Notes

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1) (408) 441-0311
Fax: (+1) (408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Roa
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel München GmbH
Business Campus
Parkring 4
D-85748 Garching bei München
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81) (3) 6417-0300
Fax: (+81) (3) 6417-0370